January 2006

Deutsche Bank's TaskMan System and the Job Execution Framework

Justin Zipkin
Worcester Polytechnic Institute

Matthew R. Piette
Worcester Polytechnic Institute

Project Number: MQP MXC-W054

Deutsche Bank's TaskMan System and the Job Execution Framework

A Major Qualifying Report: submitted to the faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the

Degree of Bachelor of Science

by:

Matthew Piette
mpiette@wpi.edu

Justin Zipkin
zip@wpi.edu

In cooperation with:
Deutsche Bank

December 17, 2005

Professor Michael Ciaraldi, Major Advisor

Abstract

The goal of this project was to aid Deutsche Bank in the migration from a deprecated time based scheduling system to a new service-oriented Job Execution Framework. Throughout the project we identified shortcomings that plagued the old system and designed processes to avoid reinventing these problems in the replacement framework. We also created a system to visually represent the running jobs in the old system, and created a methodology around it for the migration of tasks to their new framework.

Acknowledgments

First and foremost, we would like to thank Greg Friel for sponsoring this project. Without his support and enthusiasm, we would not have had this opportunity to live and work in London at one of the world's top investment firms. We would also like to thank Professors Art Gerstenfeld and Mike Ciaraldi for all the time they devoted to organizing and advising this project.

We would like to acknowledge our colleagues at Deutsche Bank who helped make this project successful. In particular, Ian Ramsay, who took the time to help us get acclimated to our new surroundings and aided us in getting the project off the ground. We'd also like to thank John Hawkins for sharing with us his vast knowledge of the inner workings of the Deutsche Bank applications and technical infrastructure. Special thanks go out to Vilas Hirani and Owen Davies for taking the time to answer our many questions about TaskMan and its supporting applications.

Finally, we would like to thank all of GES IT for the support and assistance they gave us during our stay here: Andy Manton, Steve Mainstone, Brian Keogh and Nick Sharp. Thank you all!

Executive Summary

Deutsche Bank conducts thousands of trades per day in its Global Exchange Services (GES) division alone. In order to get vital information to those who need it most, the TaskMan job scheduling system is used to execute over 1600 jobs per day. Since TaskMan's inception in 1998, however, the company has grown exponentially and this scheduling framework now needs to be replaced. The replacement system, known as the Job Execution Framework (JEF), is in the early development stages and still needs to be molded to implement all of the functionality that TaskMan currently offers. Additionally, JEF aims to be expandable by utilizing a Service-Oriented Architecture (SOA) coupled with an Enterprise Service Bus (ESB). This will result in a much more robust architecture than TaskMan. A three stage, in-depth analysis of the TaskMan system was conducted, the purpose of which was to determine the best path to implementing the replacement system, as well as to suggest design decisions.

Phase one of this process focused on the analysis of all jobs within the TaskMan software. This analysis also included over 400 jobs run by similar, but less heavily used, scheduling systems around Deutsche Bank, such as RANTask and the Sydney scheduling Perl scripts. During this analysis a series of useful reports was created. These include lists of disabled tasks, explanations and analyses of job types, and an in-depth review of all TaskMan Stored Procedures. After a complete study of the three systems was conducted, it was discovered that forty-one TaskMan job types fell under four abstract categories that should be implemented in JEF: file manipulation, SQL procedure, e-mail, and batch or other customizable process.

During this phase, preliminary research into the transition of jobs from TaskMan to JEF was also conducted. Specifically, suggestions were made for how jobs can be represented in the new framework. For example, a clear mapping of all current TaskMan job types to the proposed JEF job types was decided upon. A sampling of user stories has also been included to aid in the development of appropriate business process templates and to give a better understanding of the flow of data. These user stories are a method of the extreme programming (XP) process, which will allow this project to have shorter iterations during development, as well as aid in the creation of acceptance tests. These automated tests will result in a higher quality of code without sacrificing valuable time.

The middle phase of this project was dedicated to the creation of the TaskMan Event Calendaring System (TECS). This system is a web-based application, consisting of a back-end MySQL database and front-end PHP web interface. TECS was constructed in order to provide detailed information about tasks within the TaskMan system and to ease migration of jobs to JEF. TECS can also aid support staff in situations that necessitate the manual shutdown and startup of select tasks, such as when certain markets close during the holiday season. This assistance is accomplished by the calendaring system through its ability to rapidly and intuitively visualize job structures within TaskMan. While TECS will be particularly helpful in the transition from TaskMan to JEF, it is suggested that JEF natively incorporates similar functionality to that of TECS rather than requiring the use of an external application.

While designing TECS, ease of use without sacrificing functionality was a primary concern. A short development cycle led to certain sacrifices such as a simplified user

interface and data passing mechanism. However, this proved to have negligible negative impact on the final product. Some of the key features that were implemented in TECS include:

- A simple and intuitive calendar based interface.
- Report-type breakdown.
- Detailed Day and Task views.
- Extensive report generation capabilities.

Perhaps the most important feature of TECS is its extensive report generation capabilities. Options for generating reports as CSV files, XML files, and Microsoft Project compatible Gantt charts are all included. The Gantt charts in particular allow the user to view a visual representation of running jobs in an easy to read manner. Gantt charts can be configured to generate in a format for easily determining job times, job dependencies, or even creating a chain of dependencies centered on a single task. These options are all well documented in this report, as well as in the application itself.

Even during the development phase TECS was able to prove its usefulness. The creation process of TECS led to the discovery of several discrepancies in the TaskMan system. These errors range from cyclic dependencies, to jobs waiting on disabled and nonexistent tasks. Multiple jobs were also discovered that have stop times that occur before their start times. Jobs of these types would never run, leaving them to clutter up the system. During the migration process, TECS' ability to visualize task 'trees' will also greatly aid in finding any other logical errors within TaskMan.

The final stage of the TaskMan analysis process focused on the input and maintenance of TaskMan jobs, as well as future-proofing JEF and its work-request system: the Global Incident Management System (GIMS). While the steps taken to 'future-proof' each application are different, there are several key areas that were analyzed throughout the process. As a whole, JEF will suffer if GIMS requests cannot provide sufficient information for support personnel to respond appropriately. Also, as JEF is meant to be a superior replacement application, not only must the remaining problems that plague TaskMan be corrected, but ways to prevent other problems from cropping up in the future must be guarded against.

GIMS is the key support application that manages incidents related to the TaskMan system, and is planned to be used in conjunction with JEF as well. By extracting all the entries in the TaskMan GIMS for the 3rd Quarter of 2005, a fairly broad view of the distribution of requests was acquired. During the analysis of these entries it was discovered that while there are predefined 'Fault Categories', the requests would actually fit better into a different set of groupings. Additionally, the unstructured style of entering a request into GIMS resulted in a large variation in the quality of data received for each request. By reconstructing the 'Fault Categories' for JEF's GIMS and developing a template to enhance the data collection process, a consistently higher quality of data can be collected, resulting in a system that can allow problems and requests to be handled in a more streamlined and effective manner. To this effect, a sample template has been generated, as well as a preliminary set of overhauled 'Fault Categories'.

Another problem with TaskMan is the fact that if a newly created job was based off of another, there was no easy way to identify this. This makes managing similar tasks that follow identical templates difficult to group together. By tightening up the GIMS process, the buildup of duplicated and unused jobs can be reduced greatly. By standardizing the type of data that is submitted with each request, a high-quality history can maintained, allowing for useful reporting features to be employed. Giving support personnel the ability to view a 'History' of requests within the GIMS system will also increase worker efficiency, and assist in the prevention of duplicated jobs. Additionally, by using this history to require that running jobs be reconfirmed by the requesting department at set intervals of time, the amount of duplicated or deprecated jobs can be efficiently reduced.

As portions of TaskMan's functionality will be assigned to JEF soon, the creation of a methodology detailing the transfer of jobs from TaskMan to JEF was of vital importance. Through the use of knowledge gained from the first stage of the project, as well as the use of TECS, a series of solid principles were developed upon which this methodology is based. The proposed system of classification for all the jobs within TaskMan will allow for a rapid and safe transfer of functionality from the old system to its successor in the future. By migrating tasks based on their dependencies and complexity, portions of task trees can be easily moved to JEF and run, while the remaining portions can be run from TaskMan seamlessly. Running both systems in parallel is part of Deutsche Bank's prescribed plan, and running them in this manner allows for a highly redundant system during the normally unstable process of testing and implementing a new application of this scale.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

As Deutsche Bank continues to grow as a company, the technological infrastructure must also grow to keep up with industry requirements. This report will focus on an in-house software product at Deutsche Bank known as TaskMan and its replacement: the Job Execution Framework (JEF). TaskMan is in charge of the majority of job scheduling needs within the Global Exchange Services (GES) IT department of Deutsche Bank, and as of this writing is running more than 1600 miscellaneous jobs every day. These jobs consist of moving data, transforming files, creating and delivering reports, and many other house-keeping operations. Through a great deal of research, tests and analyses, this project presents the current state of TaskMan in detail. Information such as its development history, functions, and shortcomings are elaborated on. Additionally, features of the current TaskMan system are compared to those of JEF.

As this project was conducted over a strict period of eight weeks, the research was broken into three discrete phases. Each phase is represented by a separate chapter and sometimes a deliverable. Also, each phase refers to its own set of appendices, provided together at the end of this report.

The first phase analyzed the types of jobs run by the TaskMan system. It considered how jobs are logically classified within the program, and how they could be better classified. Additionally, phase one includes research into the specific jobs run by TaskMan.

Phase two of this project focused specifically on creating and implementing a software deliverable known as the TaskMan Event Calendaring System (TECS). This section includes design decisions made when developing TECS, and a user guide for its use. Provided with this project is the source code for TECS, a task diagram, and job discrepancies that were discovered during development.

The final phase of this project focused on job maintenance, job input, and a methodology for transferring TaskMan jobs to JEF. Additionally, the process for logging TaskMan incidents was carefully analyzed. This included all aspects from conception all the way through actualization.

Chapter 2: Background

The intent of this chapter is to provide the reader with a greater sense of this Major Qualifying Project's (MQP) goals. To successfully accomplish this, the material provided in this section is presented in a broad fashion, introducing concepts that different audiences may not be familiar with. Firstly, a couple of basic financial concepts will be acknowledged, as well as a very brief history of Deutsche Bank. The financial concepts presented will be limited to basic futures and options trading, and the role that Deutsche Bank has within that industry. Secondly, important technologies related to job scheduling at Deutsche Bank will be introduced. This includes industry standard technology, as well as in-house software used specifically by Deutsche Bank.

While the majority of this report will delve more deeply into the concepts explained in this chapter, it is important to have an overall understanding of the technologies in use. Additionally, although not completely crucial, a solid understanding of the financial aspects in the banking industry will help clarify portions of this report.

Exchange Traded Futures and Options

Basic Services

A futures contract is a standardized forward contract that is traded on a futures exchange. This means that it is a standardized contract to buy or sell a certain asset at a pre-agreed future point in time. The exchange acts as counterparty to all trades and sets the margin requirements. In many instances, options are traded on futures. A 'put' is the option to sell a futures contract, while a 'call' is the option to buy a futures contract. In either instance, the 'strike price' is the futures price at which the future is traded, if the option on it is exercised (Veale, 2001).

The Life Cycle of a Futures Trade

Each trade has a definitive start and end point, and what occurs between these points is what is known as the lifecycle of a trade. Futures are often traded amongst investors for the duration of their life, and the responsibility of a futures contract can be transferred to many different people prior to its expiration. When a future finally does expire, the holder is simply responsible for either collecting, or paying the losses of, the difference between the entry price and settlement price. It is nearly unheard of for an investor to purchase a contract with the intent of obtaining the goods contracted in a future, and the hope is that the settlement price will yield a gain for the holder (Ian Ramsay, Personal Communication, 2005-10-24).

Deutsche Bank

Company History

Deutsche Bank has come a long way since their humble beginnings in Berlin in 1870, expanding to encompass about 964 billion Euros worth of assets, and retain nearly 64,000 employees. They are currently one of the world's leading international financial service providers, with branches in 74 countries. The institution has five core businesses:

Global Markets, Global Banking, Private & Business Clients, Private Wealth Management, and Asset Management (Deutsche Bank, 2005). This project deals primarily with Global Markets and Banking, and those are the areas that will be focused on for its duration. Specific attention will be given to the Futures and Options business.

Global Exchange Services at Deutsche Bank

Global Exchange Services (GES) is a sub division of the Operations, BAC, and Infrastructure IT department (OBI IT). GES provides comprehensive facilities to support execution and clearing of listed derivatives on all global exchanges. By providing access to a broad range of electronic execution and clearing facilities, they further simplify the process for their consumers.

All applications provided by GES are designed and supported by the Information Technology (GES IT) division. This same division has been responsible for maintaining and modifying a software product known as TaskMan, the focus of this report. GES IT is now also tasked with developing its successor (John Hawkins, Personal Communication, 2005-10-24).

RANsys and RANbase

For day-to-day storage of its many GES related transactions, Deutsche Bank utilizes RANsys, a collection of servers purchased in-house, and maintained by Rolfe and Nolan (RAN) at one of their datacenters. RANsys is a back office processing system used primarily for exchange traded futures and options. It is a flexible system that has the ability to cover global markets 24 hours a day, and is completely functional in all currencies (Rolfe and Nolan, 2005). At the end of each day, all data from the RANsys servers is transferred to the RANbase system for persistent storage, via Fobkin or COOL SQL and FTP tasks run on a regular basis. RANbase is used for long-term storage of important data. Data is typically stored for at least a year on the RAN servers in their datacenter. More important pieces of data can be stored for even more years, by being passed to servers dedicated to even longer term data storage (John Hawkins, Personal Communication, 2005-10-31).

Global Incident Management System (GIMS)

The Global Incident Management System (GIMS) is an application used throughout Deutsche Bank globally to log application incidents. The types of incidents that are logged range from feature requests, to mission critical problems and general inquiries. GIMS is the incident logging application of choice for Deutsche Bank and is likely to be used for future applications developed by GES IT.

TaskMan

History

TaskMan is an in-house scheduler application, created circa 1998 to run 8-10 Structured Query Language (SQL) scripts in sequential order. The jobs[1] performed by it are defined within a specified configuration (TKL) file which holds all of the information necessary to complete the task. It was developed and is maintained using Microsoft C++ compiled to run under the Windows XP and Windows 2000 server environments (OBI Support Group, 2005).

After the initial implementation of TaskMan, word of the software's functionality spread to other departments and other scripts and reporting features were added to it. Every time a new type of job was identified, TaskMan was modified on an ad hoc basis to be able to run that job. This type of piecemeal development continued until 2004, at which point TaskMan was populated with approximately 2500 jobs[2] (Ian Ramsay, Personal Communication, 2005-09-23). At this point, official development work related to TaskMan was frozen. Jobs were (and are) still added on an almost daily basis, but no new functionality is being built in to TaskMan.

Architecture

Although TaskMan exists as a single piece of software, it is dependent on multiple other applications to function. It is directly linked to an Oracle database where it stores and reads successfully executed jobs. This database is extremely important for TaskMan's "wait-on" functionality, which allows it to execute jobs in sequence and fail if a preceding job has not successfully completed.

In addition, TaskMan is tied very strongly to the Rolfe and Nolan RANSys and RANBase databases, as the initial purpose of the application was to transfer data between these two systems. Figure 1 on the next page gives a good layout of how TaskMan communicates with its surrounding systems.

[1] The operations that TaskMan executes are generally referred to by users as 'tasks'. We will call them 'jobs' in this document simply because the replacement system (JEF) that is to replace TaskMan will call them 'jobs'. In JEF a 'task' instead refers to the building blocks of a 'job'.

[2] While TaskMan at the time of this writing contained somewhere in the vicinity of 2500 jobs, only between 1800-1900 were enabled and running. The others are all either deprecated, or function as place-holders in the TaskMan software.

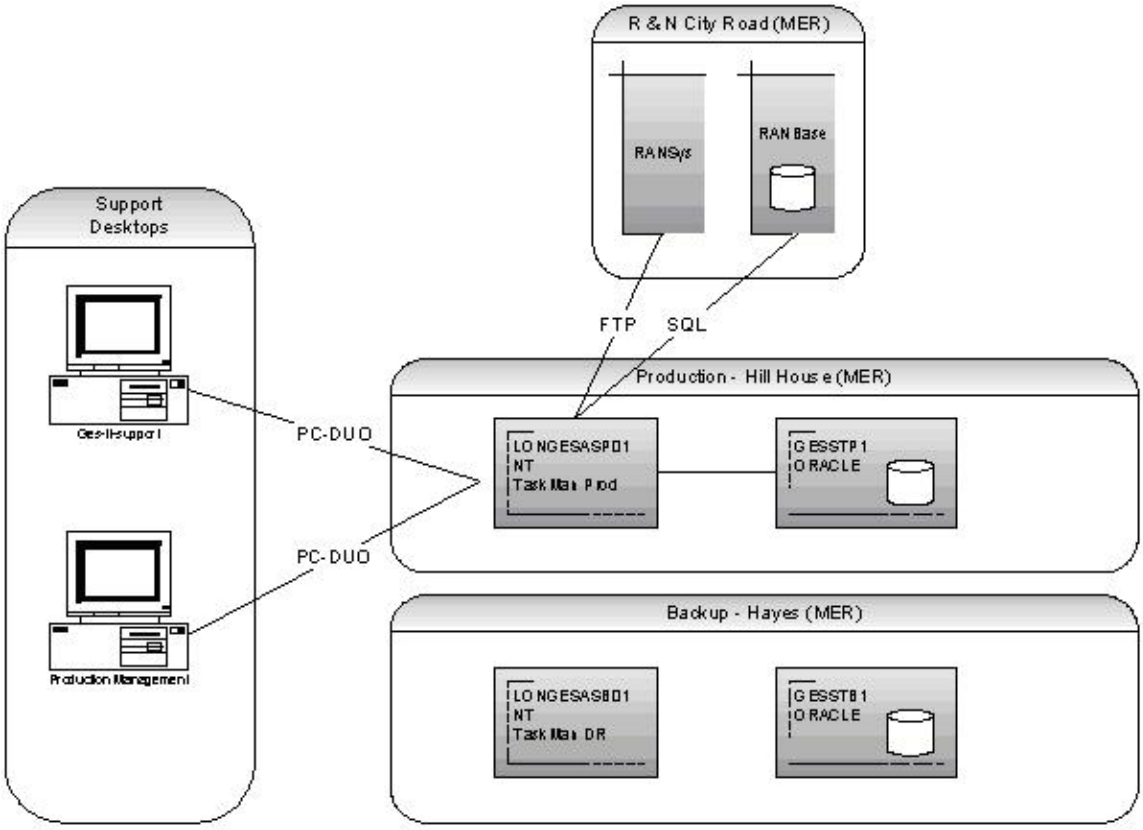

Figure 1: Overview of TaskMan in the U.K.

(Obtained from: OBI Support Group, 2005)

Problems

Although TaskMan was a good solution at the time of its development, it was not designed with the current situation in mind. TaskMan lacks various features and architectural characteristics that would allow it to excel in the ever growing environment that it is currently housed in. It is important to catalog the shortcomings of TaskMan related to its current situation if it is to be replaced by a more suitable solution.

Over the years, many people have added to or modified TaskMan in order to make the program fulfill an ever increasing array of responsibilities. Although each alteration of TaskMan was developed in accordance with current IT standards in regards to design, coding, and testing, each of these changes was self-contained. That is, each change was made without an overall design strategy in mind. As a result, the existing code base has become monolithic in nature, with very few personnel truly understanding the whole of it. Due to this, the creation of new jobs, as well as modifications to old ones have to be done by experienced Information Technology (IT) staff that have specialized knowledge of TaskMan. These factors have resulted in increased company costs and the slow down of the business handled by the scheduling system. (Owen Davies, Personal Communication, 2005-10-25).

In addition, there are numerous operations that TaskMan performs very regularly, but in a roundabout manner due to the lack of supporting code. Some of the most notable shortcomings are the following:

- Lack of SFTP or SCP support
- Lack of E-mail support
- Lack of encryption support
- A purely time-based execution cycle (no event triggering)
- A weak scheduling system (e.g. impossible to exclude specific dates)

Retiring TaskMan

Although TaskMan has enjoyed a long and successful life, the time for its retirement is fast approaching. As a new reporting system is implemented, TaskMan will slowly be phased out, and ultimately become completely deprecated. The timeframe for this is uncertain as the new system is still in the development stages, but near-production level testing is expected to start for certain pieces as early as the end of 2005 (John Hawkins, Personal Communication, 2005-12-12). After all of TaskMan's functionality is implemented in the new system, dual usage of both will most likely continue for a certain period of time, until the new system has proven itself to the Deutsche Bank staff (Ian Ramsay, Personal Communication, 2005-10-25).

Job Execution Framework

As it became apparent that the current system had a number of failings that were costing Deutsche Bank both time and money, the project to develop a superior replacement system for the aging TaskMan began. Initiated in April 2005, the TaskMan replacement project, known as the Job Execution Framework (JEF), aims to create a comprehensive framework for scheduling, job execution, monitoring and workflow. By building a framework rather than a monolithic system, the company aims to incorporate bank standard technologies where appropriate and allow for inclusion of future bank developments (Operations, BAC, & Infrastructure IT, 2005).

Service Oriented Architecture

Deutsche Bank's Job Execution Framework will be constructed using a design concept known as Service-Oriented Architecture (SOA). An SOA consists of a collection of services that communicate with each other. The communication can involve either simple data passing between two services, or it could involve several services coordinating some activity. For example, a Transform service could call a Pickup service to acquire a file, run the Transform on it, and then finally coordinate with the Deliver service to put the changed file back (Steve Mainstone, Personal Communication, 2005-10-28).

Typically business operations running in an SOA consist of a number of invocations of these different components, often in an event-driven or asynchronous fashion that reflects the underlying business process needs. Service-oriented architectures are popular for a number of reasons. They clean up interfaces between

components and enable composite applications to be assembled from a mix of existing and new components. SOAs also adapt naturally to a business process approach and can dramatically improve reuse of many software solutions (Sonic Software, 2003).

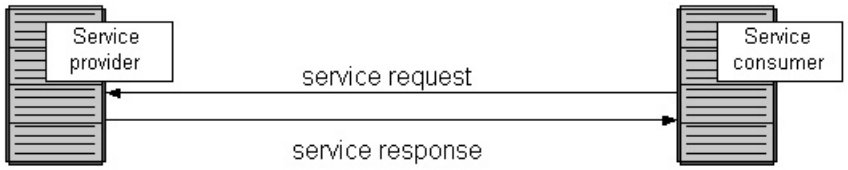

Figure 2: Service Oriented Architecture

(Obtained from http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html)

Enterprise Service Bus

The most common infrastructure used to enable an SOA is known as an Enterprise Service Bus (ESB). This is a category of middleware infrastructure products or technologies, based on Web services standards that enable a service-oriented architecture via an event-driven and XML-based messaging engine. This definition is deliberately vague, as ESBs vary to one degree or another. Despite these differences, there are some key things to look for when finding the correct ESB for a specific application.

In Deutsche Bank's situation, the key areas of concern are robustness, scalability/performance, security, and breadth of connectivity (John Hawkins, Personal Communication, 2005-10-27). A robust ESB will ensure problems do not occur often, and when they do, that they have little or no impact on the service of the bus. Scalability is critical in this operation, to provide an extensible, adaptable platform for future growth. In the past, as news of the efficiency of TaskMan spread, the number of jobs it handled grew rapidly. A similar situation must be planned for, and such growth taken into account. Sufficient scalability will permit performance to remain stable, allowing operator satisfaction to remain high. When dealing with a large amount of finance related tasks, like Deutsche Bank does on a daily basis, security is a prime concern during both the implementation phase, and the actual day-to-day operation of the final framework. Finally, as JEF will need to interact with not only a wide variety of worldwide databases, but also with a variety of file types, the extent of connectivity within the ESB is crucial to a successful final integration (Cragg, 2003).

Keeping all these factors in mind, Sonic ESB has been selected for use by the Deutsche Bank development staff. It utilizes a service-oriented architecture, which is based in the standards that the development staff is using, such as XSLT and several Java standards. Additionally, it is highly scalable and provides a fast response time at a lower hardware cost. Finally, Sonic ESB has a very flexible security infrastructure that allows for a wide variety of necessary security protocols to be implemented (Sonic Software, 2005).

It is possible to adopt an SOA approach to integration without using ESBs. There are various forms of Enterprise Application Integration (EAI). This is the use of software and architectural principles to unite a set of enterprise computer applications. The most popular method at the time of this writing is to use a bus to connect numerous separate

systems together. Using a bus architecture allows developers to connect several services over one pathway without creating multiple “point-to-point” tunnels. Aside from using buses, connecting at the database level or at the user-interface level are the most viable alternatives. However, they are not widely accepted as the "best" way, and will not be used in Deutsche Bank's Job Execution Framework.

Control-M

While JEF will be built on an event-driven architecture, it will need an additional program to schedule its batch processes. To do this, Deutsche Bank is using BMC Software’s Control-M, “a business integrated scheduling product that focuses on the production environment's business applications and platforms. It provides advanced production-scheduling capabilities across the enterprise from a single point of control” (BMC Software, 2001).

By utilizing this application as a "front-end" for JEF’s scheduling needs, Deutsche Bank will not have to develop a new piece of in-house scheduling software. There is no need to reinvent the wheel, as the batch processing capabilities offered by Control-M are exactly what are needed to execute tasks on JEF efficiently. Specifically, Control-M will be used to schedule “triggers” which will direct JEF to begin the execution of a stored job. This produces many benefits, which include cost-savings, a shorter initial development phase, and most importantly, easy support for new processes (Personal Communications, John Hawkins, 2005-10-27).

Chapter 3: Methodology

While the overall goal of this report was to assist Deutsche Bank in the migration from their legacy TaskMan scheduling system to the new Job Execution Framework, this required research into several technologies utilized by the GES IT department. Software that we considered throughout our stay included TaskMan itself, RANTask, the Sydney scheduling Perl scripts, RANBase, RANSys, and GIMS. We analyzed approximately 2500 scheduling jobs, and identified specific features that should be paid attention to during the migration. Additionally, we have presented the GES IT team with tools to facilitate re-obtaining the newest versions of the TaskMan data that we have collected with minimum effort.

To make optimal use of our eight week stay, we decided to break our project up into 3 discrete phases. Each phase was broken into two week periods, and we allowed an additional two weeks at the end for cleaning up pending research and finalizing our report. Each phase is described in detail in the following sections, giving an in-depth explanation of our methodology.

Analyzing TaskMan's Job Types

Analyzing the types of jobs supported by TaskMan was crucial in understanding the functionality provided by the system. Unfortunately, because many TaskMan job types were added throughout the history of the application's development lifespan, there appeared to be no documentation that explained the details of each job type. In addition, we found that TaskMan's user interface did not provide a simple means of collecting detailed information for multiple jobs. This made it difficult to perform any sort of useful analysis through the use of TaskMan itself. Fortunately, TaskMan stores its database of scheduled jobs in a single specially formatted binary (.tkl) file per instance. TaskMan then offers a command line switch for translating a tkl file into a legible ASCII document. This was a key discovery, and for the duration of our research we based our findings almost exclusively off of the exported TaskMan tkl files, with only the occasional need for ever referring to the database through the TaskMan application.

Our first job was to decipher the export files that we had available to us, and perform varying stages of analysis on them. To do this, we created a Perl script that allowed us to convert a series of export files into a single comma separated value (CSV) file. By having the data in a comma separated format, we were able to import it into Microsoft Excel to perform our analysis. Through modifications in our original Perl script, the usage of some UNIX command line utilities such as grep and gawk, and the sorting facilities in Excel we were able to successfully determine characteristics of TaskMan jobs such as:

- Which jobs were running
- Which job types were in use
- How many of each job type was used
- Where specific types of jobs were running from

After generating several different spreadsheets for our data, we started identifying the purpose of each job. To accomplish this we spent some time analyzing jobs, and a great deal of time interviewing experienced Information Technology personnel who were familiar with the system. Their insight and observations, coupled with our newfound data of which tasks were in use, aided us in classifying the basic groups of TaskMan job types.

With this information, we were then able to generate a distribution table of jobs by type. In analyzing this table, a much smaller list of general job types was generated. These general types were based on:

- Similarities between TaskMan job types
- The numbers of each type
- Basic functionality of each type
- Functionality that was essential, but missing

These in turn were translated into 'Generic Job Types' for JEF, keeping in mind the areas that TaskMan was lacking in, as well as future expandability of the framework.

Once potential Job Types for JEF were classified, a number of user stories were created to aid in the actualization process. A user story is simply an event or chain of events that a user may carry out, written from a user's prospective. Stories were created for each proposed JEF Job Type, as well as stories that combined types to various degrees. Using this method to define requirements is a method of extreme programming that is being utilized by GES IT to produce code of superior quality in a rapid manner.

Event Calendar

After analyzing the types of jobs that TaskMan provided, we moved on to analyzing the jobs themselves. While we had initially considered generating a methodology for analysis and analyzing each job individually, this was quickly dismissed as unreasonable considering the amount of time and resources available to us. It would have involved sorting through every job, as well as extensive contact with numerous GES IT personnel, support personnel, and the operations department whom was in the middle of a major reorganization. A better option that we settled on was to propose a simple method for analyzing running jobs that would be useful during analysis and the migration of tasks. This method developed into what is known as the TaskMan Event Calendar (TECS).

Producing an application to display the execution of TaskMan jobs in a useful manner proved to be challenging on many different levels. There were a multitude of issues that arose due to jobs in the TaskMan system that were scheduled in an illogical manner. Additionally, we had a tight time frame in which to identify all the issues that could affect our final product and address them. In the midst of identifying and addressing these problems, we were also struggling to find a method of visualization that would be of assistance to the GES IT staff. As a result of our efforts, we wound up incorporating several types of report generation schemes into TECS, conveying visualizations that were useful in identifying different types of jobs.

Architectural decisions

Before any code could be written, an overall architecture was designed for TECS. This design was based on the functionality that the system needed to provide, which we initially defined simply as "A calendar of TaskMan events". The specific goal, though, was to give GES IT a feel for what tasks ran, when they ran, and how they depended on each other. This would be helpful to the department in a number of key ways. First, it would allow staff to easily locate "critical points of failure", jobs that, if stopped, would cause a variety of other processes to halt as well. Through interviewing members of the department, we were told it would be useful to have a way of viewing this dependency chain. This is because sometimes one or more tasks need to be stopped manually, but there is fear of interfering with other necessary processes. Also, this type of visualization can be helpful in a number of troubleshooting scenarios.

After understanding these requirements, we gave consideration as to whether this system should run as a stand-alone application or as a web-enabled application. In this case, the web-enabled application was best suited for both the environment and the users. It was clear that in general, web applications offer rapid, simple deployment and widespread availability. In addition, coupled with a database back-end, a web application can provide very easy data access and manipulation capabilities.

Technology decisions

Upon concluding that a web application was the most suitable method of implementation, we decided that a database back-end was a necessary means of information storage. This would give us easy access while writing the front-end application, and also a means of viewing data that was cleaner than the raw export files that we had been working off of previously.

In terms of immediate technological decisions there were two main factors to consider. The first was choosing a Database Management System (DBMS). In our case there were two options available: Oracle Server and MySQL Server. Oracle Server is the DBMS of choice for GES IT, and also is widely considered a more robust server than MySQL server. Unfortunately, Oracle Server requires a license per instance, and the administrative consequences of this were substantial. We would need to file a request for a new instance and wait up to a week for the request to be addressed. Additionally, this would incur licensing expenses to implement. After some consideration, we decided that MySQL server would prevent this hassle and allow development to start on the application immediately. In addition, the system would be fairly simple and would not require extensive server resources. The syntax of a MySQL SQL statement is also very similar to that of an Oracle SQL statement, and would allow any GES IT personnel who wished to extend the application to do so with minimal hassle.

MySQL	Oracle
+ Fast connections + Easy replication + Overall ease of use + Free - Fewer advanced features	+ Already in use by company + More robust - Licensing fee - 'Bureaucratic red tape'

Table 1: MySQL versus Oracle

The second major technological consideration was to choose an appropriate

scripting language in which to write the calendar and its interface. First, we considered Java Server Pages (JSP), as it provides a means of scripting using native Java libraries. This would be a desirable solution as GES IT's preferred in-house programming language is Java. This would keep TECS' technology consistent with those currently being developed in the department. Despite this fact, there were a few factors that made JSP an undesirable language. Firstly, while we have experience in Java, neither of us has direct experience with JSP. This would mean for a steep learning curve while trying to implement such a solution, especially considering the small timeframe for our initial deployment. Secondly, in the initial trial run of JSP, there were difficulties choosing a suitable Java Servlet/JSP server as well as in getting expected results from testing servers. The second consideration, and ultimately final choice, was the PHP: Hypertext Parser. The advantages for using PHP were numerous. Firstly, it is a very familiar language for both of us and would allow for rapid deployment of a usable product. Secondly, the new object oriented features of PHP 5 would allow for the code to be structured in a fashion very similar to that of a Java program. This feature would make the code more easily readable and extendable by the GES IT staff with minimal additional training. The main disadvantage of using PHP is its unfamiliar nature to the department, and this issue was minimized by following an object oriented programming paradigm, and thoroughly documenting our code.

Development

The first step that we took in developing our application was devising a method for populating our MySQL database with TaskMan information. We again decided to use the data from the TaskMan export files. We developed a Perl script capable of reading in an export file, parsing it, and then inserting the relevant information into our database. This script could easily be run on a regular basis, perhaps even as a TaskMan job, to keep the database up to date.

After we had a populated database we began working on the front end for the application in PHP. Bearing our timeframe in mind, we tried our best to develop unrelated pieces in tandem. For example, while the initial data manipulation classes were being developed, the user interface was also being developed. Although we had a good idea of how we wanted our program to fit together in terms of class interdependency and inheritance, we spent time theorizing about the architecture during the development process, and nothing was finalized until the application's initial release.

One of the least planned sections of the application during development was the report generation facilities. We had a basic idea that we wanted to be able to visually represent the running tasks, and while on one hand we knew we could easily create a "calendar view", we also wanted to produce something more detailed. In the early stages of development two export features were designed: export CSV and export XML. While these reporting capabilities were somewhat helpful, they still did not provide quite the visualization we were looking for. After a preliminary demonstration of our system's functionality to our supervisors, it was suggested that we come up with some type of visualization similar to that of Microsoft Project. At that point we began doing a bit of research and discovered that Microsoft Project had a well defined XML schema for importing Gantt charts. Being that Microsoft Project is the planning software of choice at

Deutsche bank, we proceeded to work on an export feature geared specifically towards generating Microsoft Project Gantt charts. While programming the reporting functionality, we discovered three methods for generating Gantt charts that created useful and mutually exclusive results. In our final distribution of TECS, we allow the creation of three different types of Gantt charts:

- A single day's time-based representation
- A single day's dependency-based representation
- A single tasks dependency chain

The first two types of charts show the same tasks, but allow the user to view different aspects that may be useful. The time-based representation will give an accurate depiction for the time that all tasks run on a given day. In contrast, the dependency-based representation will not accurately portray times, but will create dependency chains for all tasks that wait on other tasks for execution. The decision to implement separate report generating options for these is due to the design of Microsoft Project and its inability to represent tasks that depend on other tasks that are not yet complete. The final Gantt chart type will create a chain of dependencies based off of a single task, which is often more legible than generating a Gantt chart for an entire day.

During the implementation of our report generating facilities, we also came across several discrepancies in the scheduling of TaskMan jobs. The miss-scheduled jobs became evident during both the development phases, and while reviewing the output charts generated by TECS. Although we had not initially planned to be detailing problem tasks during this phase of the project, we have both addressed the problems with the appropriate staff and included our findings as appendices to this report.

Design Issues

There were some minor design issues that we encountered very early on in the development process. One example of this is hardships that we faced in designing a visually pleasing, easily usable interface. Fortunately, this issue was remedied in little time, and through the use of cascading style sheets we were able to easily fine tune the interface throughout the development cycle.

Technologically, the most difficult part of development was writing the report generating facilities to iterate through data in the MySQL database and parse out the necessary data in a good format. While attempting this, we discovered that our Perl script that populated our database did not take certain, previously unknown, considerations into account. This was mostly because the TaskMan export files did not always appear to be populated as expected. One particular idiosyncrasy that we found was a task that had the first "wait-on" field populated blankly, and then the second one a coherent task. This task turned out to have a cyclic dependency issue as well, but the strange order of wait-on's threw our Perl script's interpretation of the dependencies off. Additionally, the cyclic dependency confused Microsoft Project when our PHP script tried to generate Microsoft Project compatible XML. Once this was realized, we made minor modifications to our database population script and noted so in the internal script documentation.

Some other logic based challenges included discrepancies within the TaskMan system itself. There were a number of tasks that had end times that unexpectedly took place before their start times. Upon realizing this, though, it was fairly easy to account for within our program. Additionally, numerous tasks "wait-on" other tasks that are no longer set to run. By disabling a task at the "head" of a chain of other tasks, it effectively shuts down all tasks that wait-on it, either directly or indirectly (Vilas Hirani, Personal Communication, 2005-11-15). Our findings of unusual tasks are detailed in the attached appendices and can be researched in greater detail by referring to Appendix D: Task Discrepancies as well as referring to our included files listed in Appendix E: Included Media.

Job Input and Maintenance

While we spent a large portion of our time familiarizing ourselves with the TaskMan system and creating utilities to help in the analysis of jobs, the ultimate goal was to offer suggestions about the migration to JEF. Following the release of TECS, we shifted our studies towards factors that would assist in a successful JEF deployment. Before drawing any conclusions, we started at the beginning, and decided that if there was anything that needed to work well in the initial deployment, it was the job input facilities. We had already known that TaskMan was lacking in this area, because of the general lack of knowledge for what TaskMan was doing at any given time.

We decided to start by analyzing the request system for dealing with TaskMan requests, and turned our heads towards GIMS. Through communication with Owen Davies and Vilas Hirani, we sketched out the cycle of events that occurs each time a request is entered in GIMS. Doing this enabled us to acquire an understanding of how the system worked. However, a more detailed analysis would be needed in order to produce useful information. As a result, we decided to extract and analyze a portion of GIMS requests from the 2005 fiscal year.

We exported the entire group of 3rd Quarter 2005 requests from GIMS into a spreadsheet. By analyzing its contents, we were able to acquire a good insight into exactly what type of information is collected by TaskMan's GIMS, as well as the quality of this data. By organizing each request by a new set of "fault categories," a new distribution table was created. This reclassification would assist support personnel in dealing with the tasks that come in. Additionally, a standardized template was created in order to heighten the quality of data submitted via GIMS.

Another important aspect of this phase was to present suggestions for preventing JEF from inheriting TaskMan's deficiencies. The various technologies that JEF will be comprised of, such as the Sonic ESB and Control-M, were all analyzed to make sure that they lent a measure of future-proofing to the framework. We also point out specifically useful design considerations to keep in mind that we have discovered through our research.

After suggesting best practices and documentation procedures for creating jobs in JEF, we began working towards a methodology for transferring tasks from TaskMan into JEF. As JEF is supposed to be a full fledged and superior replacement for TaskMan, it will be important to have a method of moving data from system to system. There also

seemed to be interest within the department of having some sort of suggestion for how to do this. After our thorough analysis of the different types of TaskMan jobs, we proceeded to locate and categorize them based off of how they depended on other jobs and what actions they performed. We then proposed a safe method for migrating these tasks based on a redundancy principle, allowing the department to "fail back" to the proven TaskMan job in the event that there is an error with an early release of JEF. By acting as a failsafe, TaskMan can be kept in operation until it is proven that JEF will execute at an expected level of efficiency and reliability.

Chapter 4: Preliminary Analysis

TaskMan is a small and powerful application that has grown far larger than originally intended. The initial planning phases of TaskMan never accounted for a system that would be running thousands of jobs, and the architecture is not sufficient for such a scenario.

This chapter includes a thorough analysis of the current architecture of TaskMan, specifically considering the types of jobs that have been added over the past several years. To easily retrieve similar data from the TaskMan export files in the future, a series of useful Perl scripts have been written. Please refer to Appendix E: Included Media for information regarding included source code.

TaskMan Type Classifications

TaskMan currently contains 41 "pre-defined" task types. Of these 41 types only 27 are in use today. Below, Table 2 shows these types, bolding the ones currently in use. For a brief description of the non-deprecated types refer to Appendix C: TaskMan Job Types.

Posting (Old)	**GENLED**	**BSR**	**Export Mappings**	(Proposal) Archive	(Proposal) R&N Report
Premium	**Profit & Loss**	**Variation Margin**	**Cash Settlements**	**Stored Procedure**	**P&L Premium**
Exercise Premium	Import Mappings	**FFTFAX**	**Capital Adjustments**	Tidy Reports	SAPGD (run 1)
SAPGD (run 2)	**FOGL**	**Run SQL File**	**Intermeadiate IM**	**Initial Margin**	**Commission**
Volume Reports (daily / monthly)	Volume Reports (Equity / Global)	RANBase Integrity Check	**Convert File Format**	**File Import**	**Squash / Difference GENLED**
Launch Ranrec	Generate COOL 'doc' Files	**Bulk File Import**	CV Feed (LCF)	CV Feed (Customer)	Export COOL Reports
Database Table Archive	**Replix FAX**	**File Import2**	**PalRep**	**Event Monitor**	

Table 2: List of TaskMan Task Types

Of the 27 running job types defined in TaskMan, four of them comprise over 90% of the entire system. These four core types are shown in Table 3, along with their associated job count. Please note that this includes all tasks, both running and disabled. Also be aware that this information is current as of 2005-10-24.

While this method of classification makes it simple to see what types of operations are being performed, the 142 "Other" tasks and 454 stored procedures (essentially batch files) say little about what is actually being done. It is more advantageous to classify the

Classification	Job Count
Bulk File Import	235
File Import	392
SQL File	1024
Stored Procedure	454
Other	142

Table 3: TaskMan Core Types

jobs using abstract functional types. In analyzing the current system, every job type that is being used today can be functionally classified as one of the following.

File Manipulation Operation

File manipulation operations consist of jobs that perform any sort of transform on a file. A transform can be moving a file, reorganizing data in a file, adding to a file, or deleting from a file. The built-in TaskMan types that can be classified as file manipulation operations include: Bulk File Import, Convert File Format, FFTFAX, File Import, File Import2, Replix Fax, and Squash / Difference GENLED Files.

SQL procedure

In its simplest form, a SQL procedure simply takes a .sql file containing SQL code and runs it against a database server. This is how the built-in Run SQL File type works. There are also other job types in the system that are essentially just running SQL code that has been hard-coded into the program. These files could probably be rewritten as simple SQL procedures, and may be in the future. Built-in job types that work this way include: BSR, Database Table Archive, Event Monitor, Export Mappings, GENLED, PalRep, Run SQL File, and Volume Reports (daily / monthly).

Fobkin (Futures & Options Back Office Interface) Procedure

Fobkin procedures consist of code that moves data from the day-to-day RANSys database to the more persistent RANBase archive. Essentially, Fobkin is performing SQL queries itself, but they are of a special type that sets them apart from other general SQL procedures. The initial job types that TaskMan was based around existed for the purpose of invoking Fobkin procedures. The built-in TaskMan types that can be classified as Fobkin procedures include: Capital Adjustments, Cash Settlements, Commission, Exercise Premium, FOGL, Initial Margin, Intermeadiate IM, P&L Premium, Premium, Profit & Loss, and Variation Margin.

Batch Job

A batch job is a single job triggered individually that invokes one or more other operations. In TaskMan a batch job is a simple MS-DOS batch script that is custom tailored to suit a specific need that defies any one of the above job types. Many of the DOS batch scripts perform simple and common operations such as sending an e-mail, making a secure FTP transaction, or even performing standard TaskMan jobs with a special check. Batch jobs are represented in TaskMan as the built-in Stored Procedure type. Table 4 gives a more detailed breakdown of the types of tasks that occur in batch jobs.

Classification	Job Count
.CTL Linkers	114
Email	78
File creation	17
FTP	87
SQLPLUS	25
Misc	54

Table 4: Stored Procedure Composition

Asia-Pacific Type Classifications

The Sydney branch uses a UNIX cron daemon to schedule and run its Perl scripts. This is a good solution, as it does not need to run as many tasks per day as TaskMan does, and allows for great flexibility. One of these Perl scripts, ran2db.pl, acts as the main driving engine behind the remaining scripts. It does this by accepting a configuration file in the command line. Parameters within the configuration file then tell the Perl script how to execute. Tasks performed based on these parameters include:

- Log onto any R&N ftp server, using an encoded username and password
- Retrieve a file
- Place a flag on the FTP server to indicate that the trade date's data has been processed
- Poll a specified number of times before timing out
- Email the file out to a list
- FTP the file out
- Send it to a Samba mount
- SCP the file to another server
- Delete a file

While many of these parameters mimic functionality of the File Manipulation and Stored Procedure Job Types from TaskMan, SQL Procedures are not covered. This is where something known as a "hook process" comes in. This allows for the main Perl script to call other, custom Perl scripts through the configuration files. By doing this, the Asia-Pacific region can fulfill a wide range of customer needs, from custom reports to database extraction and manipulation (Vietlong Le, Personal Communication, 2005-11-03).

RANTask Type Classifications

In Frankfurt, another scheduling system called RANTask is used in addition to TaskMan. RANTask provides similar functionality to TaskMan, but makes implementing some jobs simpler. Job types in RANTask can be added by installing additional modules to custom tailor the software to companies needs. For a listing of supported RANTask types refer to Appendix B: RANTask Task Types.

Chapter 5: TaskMan Event Calendaring System

The TaskMan Event Calendaring System (TECS) is a web-based application designed and implemented as a major deliverable for this project. It consists of a back-end MySQL database and front-end PHP web interface. TaskMan task[3] information is populated into the MySQL database through a Perl script that is run against TaskMan export files. It is the aim of TECS to make the transfer of tasks from TaskMan to JEF as easy as possible through visualizing running tasks in a well-structured and easy-to-navigate manner.

Documentation

As a fairly basic web application, the only client-side requirements to run it are a current web browser, and access to the server that the application is installed on. This application has been tested successfully with both Internet Explorer 6 and FireFox 1.0.7. The computer that the application is run on must have at least MySQL 5.0.15 and PHP 5.0.5 installed, as well as sufficient hardware capabilities to act as a server.

Looking at the main screen there is a fairly basic layout, as seen in Figure 3. At the top, there is a title bar. Taking up the left hand side of the screen is a navigation bar that containing links that take the user to their desired month within the current year, and also back to the current date. Occupying the upper portion of the center of the page is the calendar itself. It contains links to the previous and next months, links to each day of the month, as well as select days from the end of the previous month and beginning of the next. These days can be easily differentiated from days of the current month, as they are shaded light grey. The blue box in the calendar denotes the currently selected day.

[3] Although this report makes an attempt to always refer to TaskMan tasks as "jobs" to be consistent with JEF terminology, this particular chapter is an exception. As TaskMan does not actually have "jobs", TECS has been programmed to recognize a TaskMan task simply as a Task. For the sake of readability, this chapter will maintain that terminology.

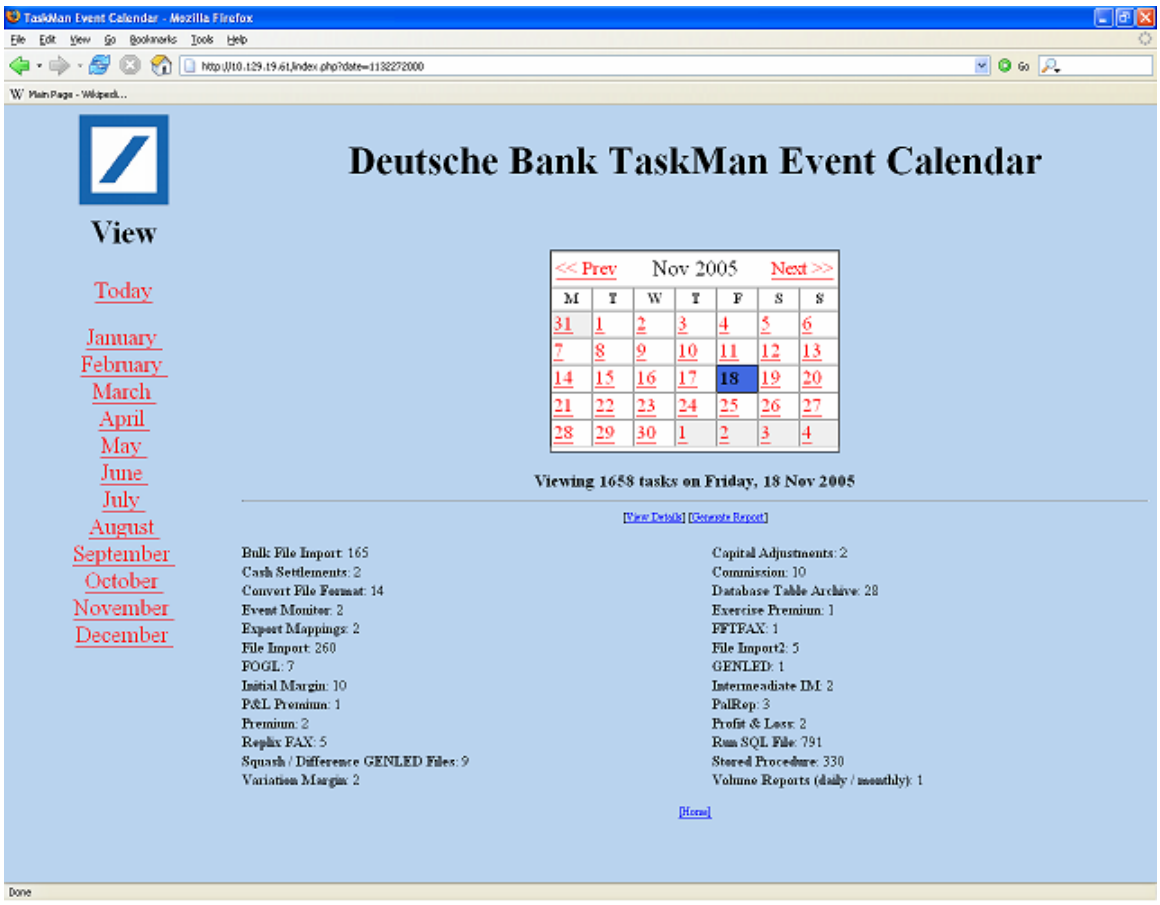

Figure 3: TECS Main View

Directly below the calendar is information denoting the current date, as well as a summary of all tasks that run on that day. Taking up the lower portion is a breakdown of the running tasks by job type. Located below the calendar and above the distribution are two hyperlinks. The “Generate Reports” link takes the user to a webpage that explains the purpose of each type of available report in detail. The four download links allow the user to save a file containing the current day’s data in a variety of formats including standard XML, CSV, or a specialized Microsoft Project compliant XML file that can be loaded into Project to display a Gantt chart. The time-based Gantt chart will show the start and end times of all tasks running on that day accurately, but will not show any task interdependencies. The dependency-based chart disregards an accurate depiction of the times in favor of a more accurate showing of the predecessors and descendants of each task. The “Home” link will take a user from the current page back to the main page.

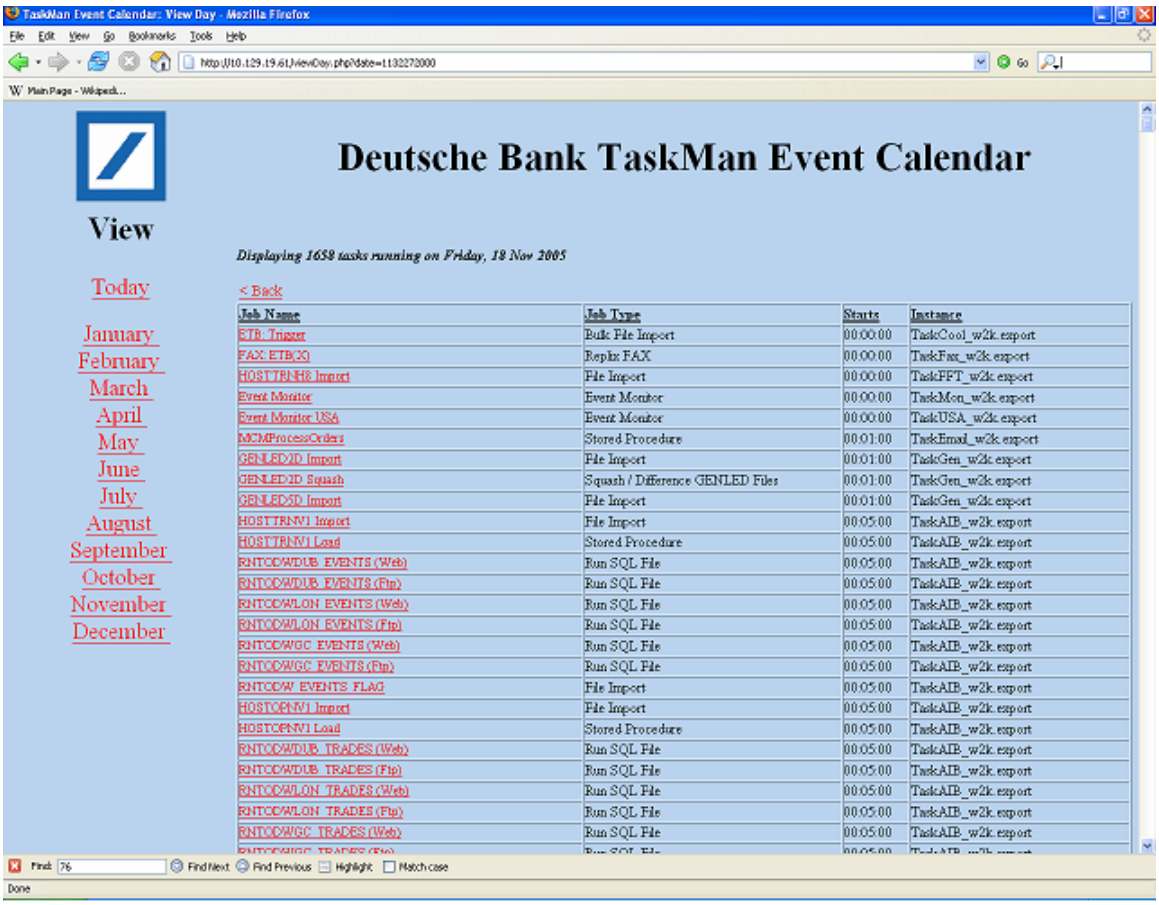

Figure 4: TECS Detail View

The “View Details” link, either on the main screen or the report generation screen, will take the user to a webpage that contains a more in-depth view of the tasks that run on the selected day as shown in Figure 4. Tasks displayed in this manner are ordered by their start times, with their names formatted as hyperlinks. Clicking on one of these will bring the user to a screen that contains highly detailed information concerning that task, including links to all tasks that it must “wait-on” in order to run, as seen in Figure 5. This allows a user to quickly and easily view a hierarchy of tasks that rely on each other.

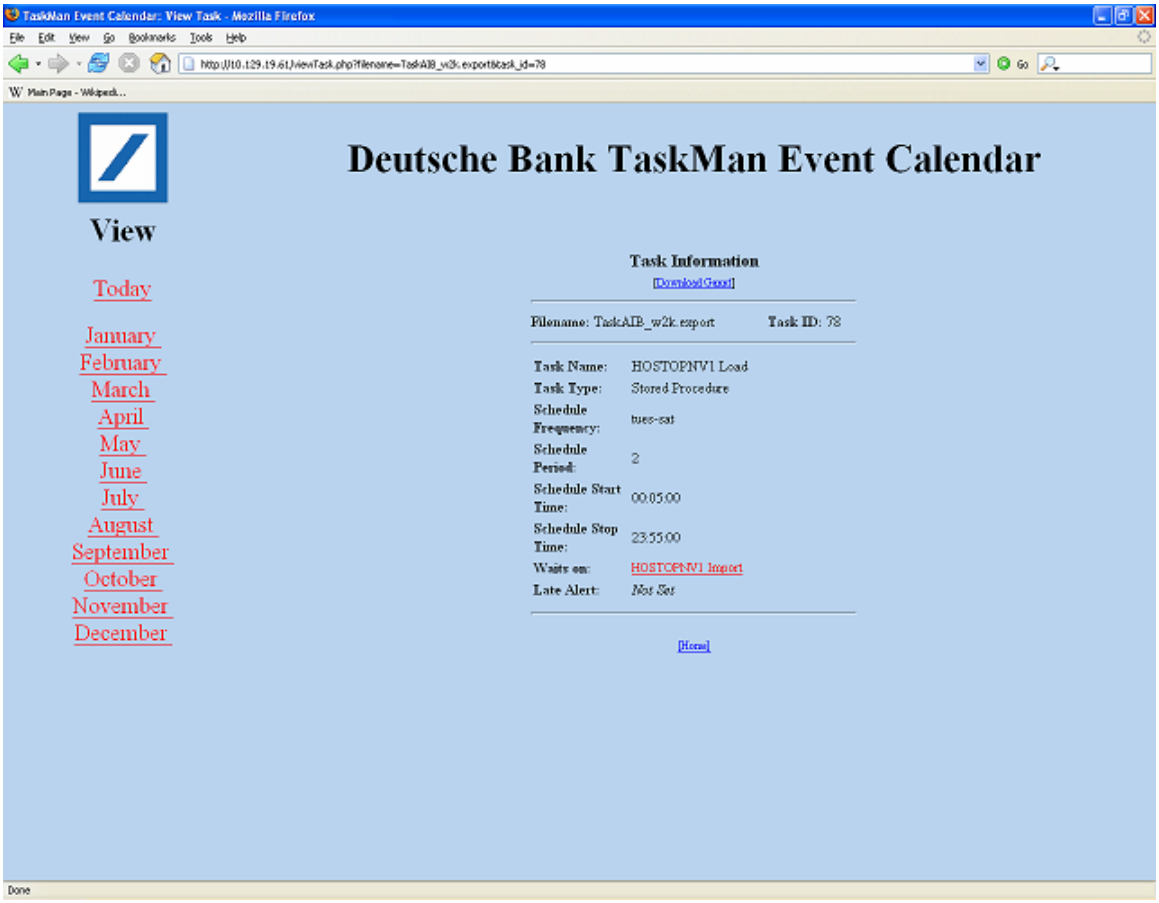

Figure 5: Task View

At the top of this screen is another Gantt chart link. Downloading this file and loading it into Microsoft Project will display a chart that details the predecessors and descendants of the current task only, as shown in Figure 6. The uppermost task in this Gantt chart is the root task. Directly below it are its predecessors, and then finally any descendants that rely on the root task to function. It is important to note that the Gantt chart will only depict predecessors of predecessors, and descendants of descendants. That is, if TECS identifies task B as a predecessor for task A, it will continue to try and find predecessors for task B, but will not try and find other descendants (other than task A) for task B. This rule holds true for descendants of task A as well, where TECS will only find decedents of the descendants, and not predecessors of them. Additionally, if the root task relies on any "special tasks", such as RANBASE EOD COMPLETE, then the special task(s) will be listed directly under the root task, followed by the predecessors and then descendants. A special task is one that is not executed by TaskMan, but TaskMan can recognize its completion. This is accomplished by some external process modifying the TaskMan database.

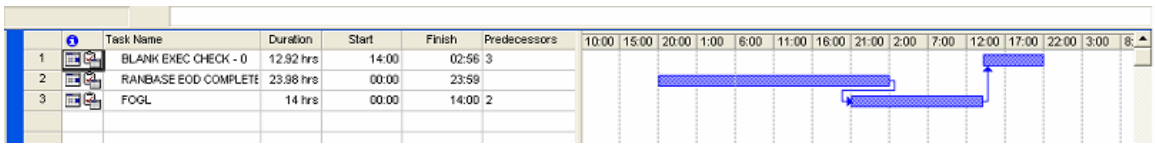

Figure 6: Example Task Gantt Chart

Chapter 6: Job Input and Maintenance

The study of the input and maintenance of jobs in a scheduling system was essential in understanding the manual labor and costs associated with a particular system. This chapter focuses on the system that is currently in place for addressing issues with TaskMan jobs: the Global Incident Management System (GIMS). In studying this system, special detail was invested into discerning shortcomings and failings within GIMS. As there is not yet a process for supporting the new JEF system, this document should be of specific interest when a request system for the new framework is implemented.

In addition to simply documenting GIMS, this chapter also makes suggestions for some implementation considerations when deploying JEF. Specifically, what types of information should be required when creating jobs, and how this can be accomplished. These suggestions have been reached through the GIMS analysis, interviews with department personnel, and research into current industry standards.

Finally, this chapter proposes a methodology for transferring tasks from TaskMan to JEF. A useful method of classification is proposed for already running jobs, and the tools for performing this classification are detailed as well.

Logging a TaskMan Request

There is currently a well defined procedure for logging a request related to the TaskMan system. The procedure involves a user, a GIMS request, and a member of support staff to analyze and react to the request. A user will file a request with the GIMS system in one of two circumstances. The first situation is if the user needs a new job created. Some users may know that they need a specific job created, especially if it is very similar to a job that they already have running in the TaskMan system. Other users may come to the conclusion that they need a TaskMan job through the recommendation of GES IT staff. The second situation where a user may file a TaskMan GIMS request is if a problem with an already running report occurs. In this instance, the user will file an inquiry using a method identical to that of a report creation GIMS. Below is a simplified version of inputting a current TaskMan request.

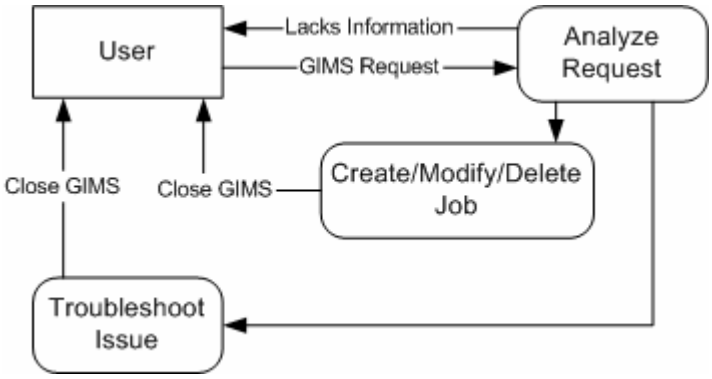

Figure 7: Logging a request

Although the above process seems quite simplistic, Figure 7 fails to show the amount of manual labor that is involved in each of the three processes.

User Input

When a user submits a TaskMan ticket, they are presented with a screen similar to that shown in Figure 8. The user is required to acknowledge all fields that are designated "primary fields", and can additionally fill in "optional fields" if they choose.

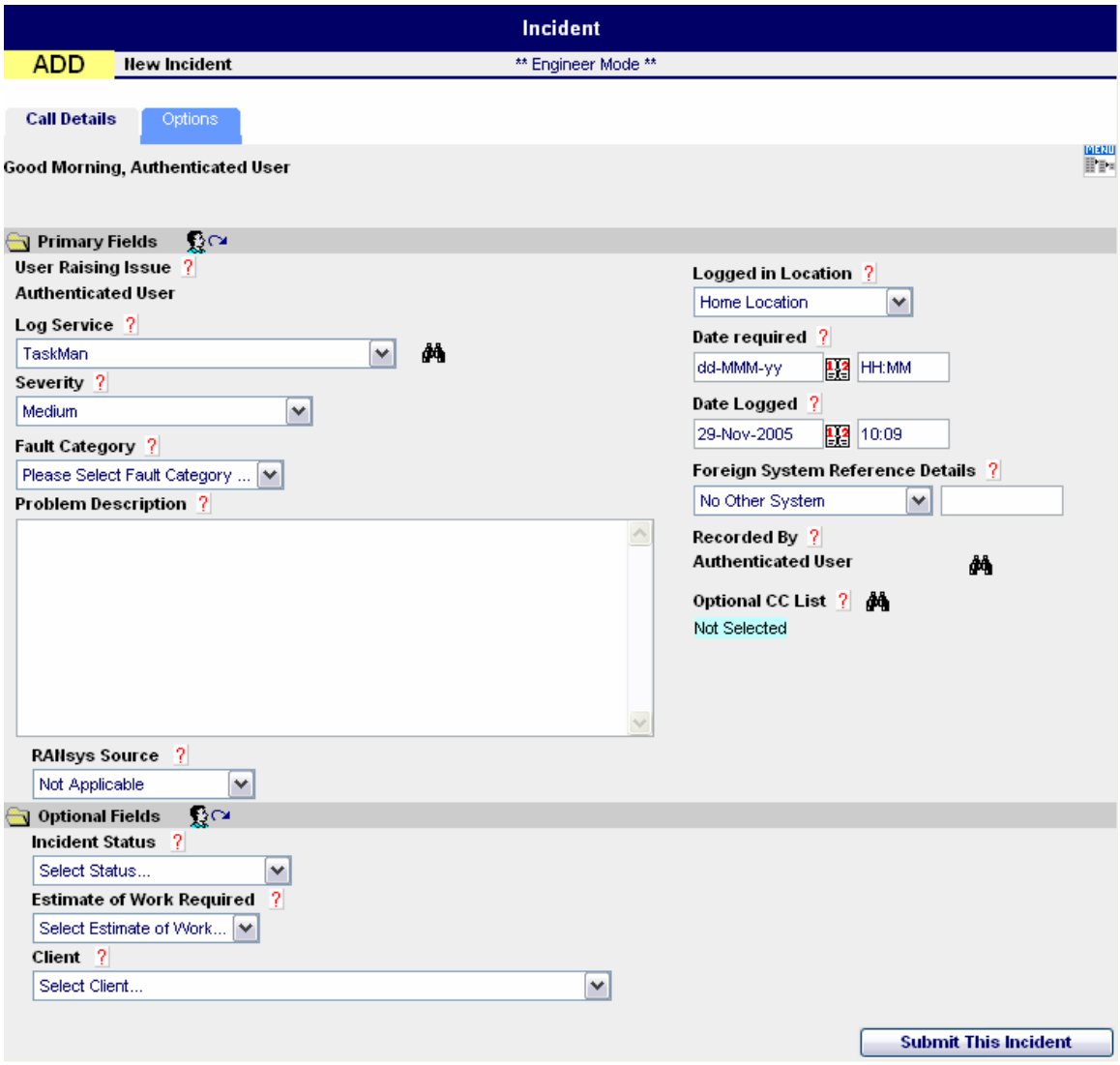

Figure 8: User view of GIMS

Severity – Choose a level of severity that the user believes their project is.
Fault Category – Choose a fault category based on the types available in GIMS.
Problem Description – A written description of the problem.
RANsys Source – If the issue is related to RANsys, the source can be selected.
Logged in Location – Determines what city your incident is routed to.
Date Required – When the issue needs to be addressed by.
Date Logged – Represents the date that the issue was logged.
Foreign System Reference Details – Only used if problem needs to be addressed by an external vendor.

Table 5: Explanation of relevant Fields

Analyze Request

One of the first pieces of information filled in by the user is a "fault category" for their entry. The fault categories available are listed in Table 6. Unfortunately, these options do not properly correspond to regularly entered requests. The deviation is so great, in fact, that very similar entries will often be entered with different associated fault categories. A more appropriate means of classification would include fault categories that account for regularly logged events.

Application Problem
Configuration
Logon Problem
LS2 Related
Static Data
User Administration

Table 6: GIMS TaskMan Fault Categories

After a request has been submitted, it is then delegated by GIMS to an experienced support staff member for analysis. This consists mainly of deciding exactly what the staff member will need to do. Realistically, a user will generally be asking for one of two things: Creation or modification of a report, or the troubleshooting of a problem.

Create/Modify/Delete a Job

If the request involves creating or modifying a report, the support staff needs to hope that the information necessary is available to them in the GIMS request. In the case of creating a job, this includes a detailed enough description to properly implement the job in TaskMan. Being that GIMS allows the user to request anything they want, without forcing them to state any explicit information, the information is not always descriptive enough. In the event that sufficient data is not available, the support staff will need to contact the user for clarification. Additionally, oftentimes the request to create a new job is based on an already existing job. This means that the support staff will essentially be digging through jobs that already exist, locating the one they need, copying it, making a slight change, and closing the case. The newly created job will also never be linked to the one it was copied from, allowing this process to happen multiple times for the same job.

In the event the user requests to modify a job, the same issue arises of there not always being sufficient information in the GIMS request to successfully make the modification. Also, once the modification finally is made, there is no formal log that the change was made, other than the archived GIMS request. This makes it hard to relate logged requests to each other if a similar issue arises in the future.

Finally, if the request comes through to delete a job, the support staff will simply go into the TaskMan program and switch the job's schedule to "off". This means that the job will remain on the main screen of TaskMan, indistinguishable from the ones currently running unless a more detailed investigation is conducted. It also means that if there are any running jobs that rely on the disabled one to run, they will stop running as well, without any prior notification.

Troubleshooting an Issue

Although a very broad category, it is quite common for a request to be made that is related to troubleshooting an issue. One common situation is when a user has not received a report that they expected to have delivered to them. This often leads to the user quickly inputting a GIMS request. It is then the responsibility of the support staff to determine why the report has not been sent out, or if it has, why the user has not received it. Generally, this process is labor intensive and can involve searching a database of completed jobs and manually relating jobs that depend on each other.

Other troubleshooting issues that are logged include requesting copies of SQL statements that are used in report generation, the creation of FTP dropboxes, and manually starting jobs that need to be rerun or have not yet started. All of these require the support staff to go in and perform some sort of action that does not follow a clear methodology.

Analysis of GIMS requests

The current GIMS system used for TaskMan job requests is hindered by the same lack of rules that has caused the scheduling system to atrophy as it has grown over the years. As it is assumed that JEF will also use GIMS as its job request system, care must be taken to ensure that this aspect of the replacement system is as strong as the rest. Having conducted a detailed analysis of the way that job requests are handled for TaskMan, a number of ways have been researched in order to rectify many of the problems currently plaguing GIMS.

Review of GIMS requests, 3rd quarter 2005

Four hundred fifty-four requests were generated by the Global Incident Management System between 2005-07-01 and 2005-30-09. As the fault category can be ambiguous, or is not always correctly selected by the person logging the request, a series of "Basic Functions" were developed to better classify the types of requests.

# of Requests	Type of Request
205	Troubleshoot a job or report
89	Modify a job's distribution
63	Modify a job's function
45	Create a job
13	File request
10	Disable/re-enable a job
5	Delete a job
3	FTP dropbox creation
21	Other Issue

Table 7: Basic Functionality of requests

As can be seen in Table 7, missing reports and problems with existing reports make up nearly half of the requests. An additional 152 requests deal with the modification of either the functionality of a job, or who receives the reports generated by

it. Approximately 10% of the requests during this quarter were for the creation of new jobs. However, many of these appeals ask to use an existing job as the basis for the new one, demonstrating that there is indeed a heavy replication of existing jobs. Of the "Other" jobs, 11 are tests, 9 are unique enough to be unclassifiable, and the last one was mistakenly submitted in GIMS. Please see the included gims_q3_2005.xls Excel Workbook for a detailed spreadsheet of these requests. Appendix E: Included Media provides a listing of where all included files can be found.

Methodology of Transferring a TaskMan task to JEF

Migrating existing jobs from TaskMan to JEF is an essential step in decommissioning TaskMan and will progress more rapidly and efficiently through the use of a well defined methodology. Rather than simply stepping through jobs in TaskMan sequentially or randomly choosing them to migrate, key factors must be taken into account. Although JEF is being designed as a robust and theoretically superior system to TaskMan, there is always a risk of error during the early stages of implementation. While TaskMan continues to function with a reasonable deal of reliability, jobs should be transitioned off gradually. Due to the flexible architecture that JEF employs, both systems should be able to run in tandem until it is clear that JEF is reliably providing all of the functionality once offered only by TaskMan.

Identifying Migratory Jobs

The first step in such a migration is classifying jobs based on their suitability for transference. For the purposes of this migration there are four types of jobs to consider:

1. Independent Jobs
2. Leaf Jobs
3. Complex Jobs
4. Root Jobs

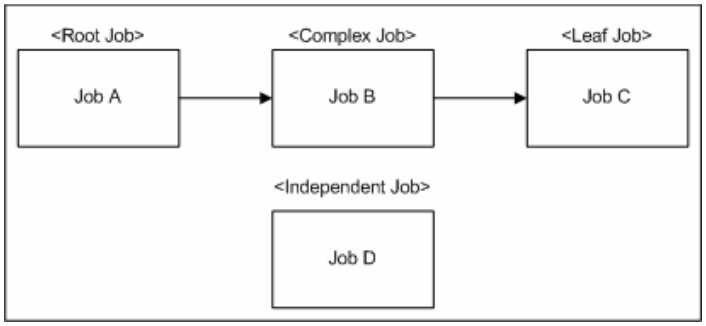

Figure 9: Job Types by Predecessor Status

Independent jobs are those that do not wait on any other jobs, and do not have any jobs that wait on them. These are particularly good candidates for early migration. Early in the migration phase, these jobs can be transferred to the JEF system and not affect any other jobs in the event of an error. Additionally, for the most essential jobs, TaskMan and JEF can run them simultaneously to ensure their completion. After JEF has run a job in parallel with TaskMan for a sufficient period of time it can be officially disabled in the TaskMan software.

Leaf jobs are those that may depend on other jobs, but have no jobs that depend on them. These are also good candidates for migration, but should be considered after independent jobs. After a leaf job is implemented in JEF, all of its predecessors can continue to run in the TaskMan system and simply send a trigger to JEF invoking the next job. This will ensure that if JEF is unable to complete a job it will not affect other jobs that are normally run by TaskMan. In the event that there is a bottleneck once the

JEF job is reached, the old TaskMan job can always be invoked forcefully until the JEF issue can be repaired. This method of running an interconnected JEF and TaskMan architecture is the safest and most reliable method for getting jobs into the new system, and is possible due to JEF's flexible architecture. Figure 10 shows an example migration of a leaf job to JEF.

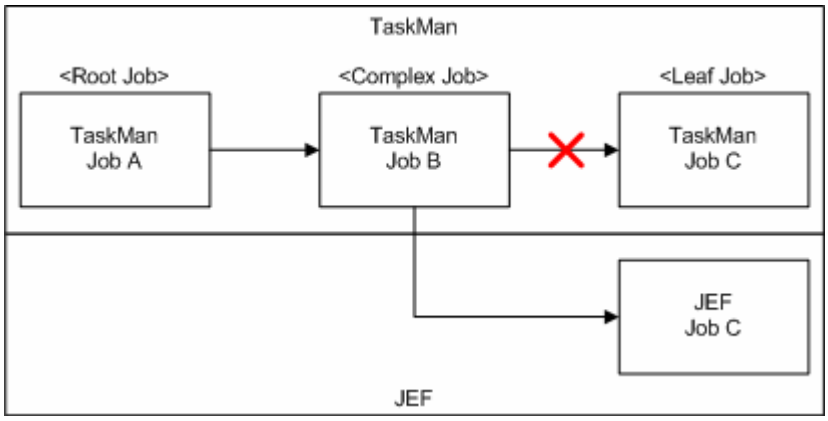

Figure 10: Migration of a Leaf Job

Complex jobs are those that both depend on other jobs to run, and have jobs that in turn depend on them to run. Because these jobs are buried deeper in the dependency tree than a leaf job, they are more difficult to migrate. Complex jobs should be migrated once all of the jobs that depend on them are first migrated. Only under special circumstances, such as lack of functionality, should one consider "skipping" a complex job in favor of another closer to the root. This prevents weaving jobs in and out of each system to the point where they become unmanageable. Figure 11and Figure 12 compare the two methodologies, crossing out the migration that is less desirable.

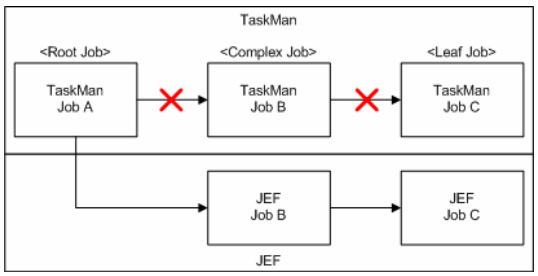

Figure 11: Migration of a Complex Job (Good)

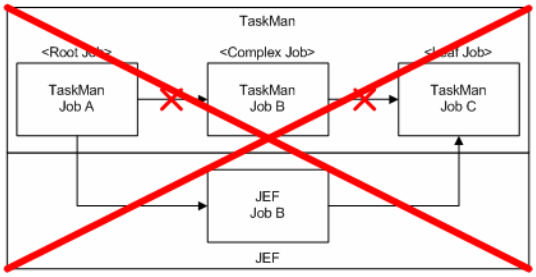

Figure 12: Migration of a Complex Job (Bad)

Root jobs are those that do not depend on any other jobs to run, but have jobs that depend on them. Jobs that are root jobs and have a great deal of other jobs that depend on them are presumably critical to some sort of operation. Root jobs follow the same rule as complex jobs where they should not be transferred until all of their dependent jobs are first transferred.

A Perl script, tm2xls.pl, was written in order to facilitate this classification. Its source code can be found included with this project. Refer to Appendix E: Included Media for a listing of file locations. This script parses an XML file, such as one generated by TECS, and from it generates an Excel file. Contained within this

spreadsheet is a listing of all running jobs, the number of jobs that each waits on, and the number that wait on it. Additionally, it detects and notes if there are any dead dependencies, jobs that appear as if they will never run.

After categorizing jobs as either an independent, leaf, complex, or root job, it is important to analyze the type of action that it is performing. TaskMan jobs that have a well defined conversion into a JEF job, or set of JEF tasks, are the best to migrate. These types of jobs include standard file manipulation operations, SQL operations, and other types that JEF has defined a template for. Jobs that will need to utilize JEF's batch functionality (similar to a TaskMan stored procedure) should be considered last, and only after the JEF development team has deemed that an expansion to the framework is not advantageous. Determining if a TaskMan job will have a well defined conversion into JEF is impossible until the framework has been fully planned and developed, but as a preliminary measure, any job that maps onto a JEF type classification other than "Batch Procedure" in **Error! Reference source not found.** of this report is a good candidate.

Although categorizing jobs into the aforementioned types is a good start for choosing jobs to migrate, there is always the possibility that circumstances will affect certain jobs in such a way that makes them poor candidates for initial transfer. The jobs will ideally be transferred by a developer who is proficient in JEF's architecture. Additionally, to ensure each task is transferred properly, an operator who is proficient with TaskMan should be available to answer any questions that arise.

Special Cases

Jobs that have a simple mapping from TaskMan to JEF should be considered for transfer early on. This should not be done blindly though. Under certain special circumstances, there may be technologies already employed at Deutsche Bank that are more suitable for a TaskMan job than JEF would be.

Jobs that are classified in TaskMan as a type of Run SQL File should have particular attention paid to them. The SQL file should be analyzed by an experienced TaskMan support member and classified as either a job that moves data, or a job that generates a report. TaskMan jobs that generate formatted reports and send them to users (such as in an XML or PDF format) may belong in a Business Objects environment as opposed to a JEF environment. This would give the user the flexibility to generate their report whenever they want or schedule it to generate regularly, as well as distribute the report as they deem necessary. If Business Objects seems like a viable option for a report, the possibility of migration should be discussed between a TaskMan support member, a Business Objects support member, and the user requiring the report.

Scheduling Criteria

After the transfer location has been determined, it must be decided whether the job should remain purely time scheduled, or should be run based on a specific event or series of events. If it will be made event-driven, determine what the triggering criteria will be. In many instances, a job's 'wait-on' criteria can be used. Numerous other triggers can also be used depending on the individual situation. Examples of these include:

- A field in a database being updated

- A file becoming available in a directory or FTP drop box
- An email being received
- A message appearing on a banks message channel
- A user clicking on a link in a web page
- A user performing an action in a 3rd party application

(From JEF Twiki, 2005-11-29)

If the job is to continue running as a time based process, it can be scheduled in Control-M to execute at a specific time. This will cause Control-M to trigger an event in JEF that will be executed through the framework. Jobs that are transferred to Business Objects can be scheduled by Business Objects' own scheduling system, and will remain completely separate from JEF. Jobs that will employ JEF's event-based scheduling feature will need to be triggered by an appropriate event that JEF will recognize.

The next step is the actual transfer of the job's functionality from TaskMan to JEF. This is accomplished by combining a suitable set of JEF tasks. These tasks are combined to form business process templates, which are then applied to specific functions to create a job. As TaskMan's jobs are first transferred to JEF, new templates will need to be constructed at a high pace. However, as the transfer process continues, fewer new templates will need to be created as previous ones can be applied to jobs of similar types.

The final step will be to fully document the new JEF job. All pertinent information should be included as part of the standard internal documentation, such as the engineer's name and department, the job requestor's name and department, as well as contact information for both parties. Additionally, the events that are needed for the job to trigger should also be listed. The priority level of the job should also be determined at this time. This can be based on a number of factors including the number of dependencies it has, the number of applications/departments affected by the job, and the frequency at which it is run. A summary of these recommended fields can be viewed below in Table 8. This information is vital to ensure a proper lifecycle for each job, as well as to allow for both a GIMS "history" and a job recertification process to be implemented.

Field	**Use**
Engineer Name	Name of the primary programmer
Engineer Department	Primary programmer's department
Engineer Contact	Email address of primary programmer
Owner Name	Name of the job requester
Owner Department	Department of the job requester
Owner Contact	Email address of the job requester
Dependencies	Criteria to be met for this job to run
Source	Source report is generated from (if any)
Priority	Level of job urgency
Date Created	Date first created
Modified	subsequent modification dates, and secondary programmer names

Table 8: JEF Job Documentation

Chapter 7: Conclusions and Recommendations

Most importantly, the replacement system that is being designed must be able to fully replace the previous systems' functionality. Additionally, it must be easily expandable to prevent a situation like this from occurring again in the near future.

JEF Type Classifications

It will be relatively easy to create modular templates that can be added to the application by reusing predefined tasks in the future. Many of the jobs running in TaskMan can easily be replaced by combining one or more JEF tasks into a template. With its initial release, there should be enough of these job templates to replace the functionality of TaskMan. This will enable the new application to achieve high levels of functionality with only a few basic job classifications. The following are the types of jobs that will be necessary to include in an early implementation of the Job Execution Framework.

File Manipulation

While TaskMan contained very useful features for file manipulation, many of these were added on an "as needed" basis. Eventually, development was frozen and anything not already specifically addressed had to be written as a batch script. JEF should contain the same functionality built into TaskMan's hard-coded file manipulation types, as well as including common operations that were used in batch scripts. The following features will be essential for a successful implementation that replicates TaskMan's functionality:

- Creating Files – On-the-fly file creation is very important. JEF should support the creation and parsing of files. Data written to these files may be either static, or dynamically generated from information read from another file or set of files.
- Deleting Files – A simple delete function, similar to delete(), as defined in java.io.file, to remove files that are no longer necessary.
- Renaming Files – A simple rename function, similar to renameTo(), as defined in java.io.file, to allow files to change names as needed.
- Moving Files – Allow the moving of files from one location or file system, to another.
- Transfer Files – Allow the transfer of files between multiple systems through an intermediate protocol. Intermediate protocols that are necessary at this point in time include:
    - FTP – The transfer of files through the standard FTP protocol (RFC959).
    - SFTP – The transfer of files through the SFTP (over an SSH connection) protocol.
    - SCP – The transfer of files through SCP (or RCP)
    - SAMBA – The transfer of files through the Samba (Windows file sharing) protocol.
- Reading Files – The reading of files for both report generation and redirection is very important. The redirection of files should include forwarding them on to an external application. JEF should also support a simple mechanism for simply checking if a file exists.

- Transforming Files – The transformation of files is essential in being able to delete, add, or rearrange information. A standard of implementation such as regular expression parsing should be implemented for such a task.

SQL Procedure

Similar to the functionality of TaskMan, JEF should also contain a simple way to run a series of SQL statements. A function that reads in a standard .sql file and executes it against a database is a must. It would also be desirable to allow for the use of variables across multiple files, and a way to dynamically generate SQL scripts. This would allow the replacement of some batch scripts, as well as the complete implementation of Fobkin procedures without having to hard-code SQL as was done in TaskMan.

In addition, many TaskMan batch types invoked stored procedures on an Oracle database server. This functionality should be added in such a way that it supports all of the features listed above.

E-mail

An area in which TaskMan was severely lacking was the ability to easily send e-mail to one or more parties. Upwards of 75 batch jobs defined in the current TaskMan system are used for the generation and sending of e-mails. This functionality should be included in the base system, either through an internal mail procedure or via an external SMTP server. An e-mail task should handle the sending of simple text based e-mail's, as well as multi-part messages in a format such as MIME.

Batch (or other customizable) Procedures

The Job Execution Framework will have tasks that defy any of the three aforementioned templates. For example, one task could need to load CAM Project Mapping tables from files and extract them again, or another could need to trigger an external application. There is no simple way to fulfill these needs easily with the given templates, and it would not be efficient to create an entirely new template for a single task. TaskMan accomplished this through the use of batch files. This was a good stop-gap measure, but would not be recommended as a permanent solution.

With JEF being developed in-house and on a Java platform, it would be prudent to allow for these types of situations without having to rely on a heap of batch scripts. Currently, the Asia-Pacific region does this by calling Perl procedures across modules. The powerful text manipulation functionality of Perl allows them to mold data to fit highly specific needs. A direct hook into Perl or similar scripting language is one solution to this problem. A custom template or perhaps the triggering/invocation of external programs for certain jobs would be other alternatives to consider as well.

Recommendations for JEF job request system

The biggest problem with the TaskMan version of GIMS is the lack of structure within the information fields. While the system has the ability to take in large volumes of data for each request, there is no set method of input. This leads to a large disparity in the quality of the information gathered. As GIMS has limited customizability, an effective, yet low-cost, solution to this would be to require submitters to simply fill out

and attach a Word template with their request. This template would contain necessary information such as requestor name, owner name, and requesting department, in addition to other basic information. This information will help eliminate confusion by leaving a more descriptive "paper trail", as well as decrease the time needed on follow-up communication to acquire necessary data that was not included with the initial request.

The standardization of GIMS requests will make the generation of a "history" of each set of requests much more viable. By allowing the engineers to quickly and easily look up all requests made about a particular job, or by a particular user, problems will be solved with greater efficiency. This will also aid in the detection of duplicated jobs, as the support personnel will be able to look up a history of requests for job creation. A suggested Microsoft Word template can be found included with this project, referenced in the attached Appendix E: Included Media. Data from this template could be much more easily interpreted by support personal and converted into a functional job. Simple operations could potentially benefit from the creation of an application that could read data out of the template and automatically perform a task. A simple example would be that of creating an FTP dropbox.

In addition to the template, the GIMS for JEF would also benefit from a new set of fault categories. The ones currently used for TaskMan do not sufficiently convey the type of each job. A set of more specific basic functions would be much more in line with the template-based nature of the Job Execution Framework. Based on the distribution of these jobs, it makes the most sense to have the following categories included: Create a Job, Modify a Job, Modify Mailing Distribution, Fix a Problem Request a File, Disable/Enable a Job, and Other. These cover the most commonly requested actions clearly, and also allow for requests that may not fall into any of the set categories.

Finally, a web-based interface that allows a client or user to view information concerning their jobs would ease the burden placed on support staff. This should be integrated into JEF to a degree that allows both the users and support personnel to easily analyze job status and issues. Many of the requests that come through GIMS for TaskMan are from users inquiring to see if a job has run successfully or not. By giving the user the ability to view information concerning their job, and possibly the option to re-run the task, this need could be greatly reduced. In turn, this would free the support staff to deal with more vital matters.

Recommendations for Applying Preventative Measures to JEF

The main reason behind the deployment of the Job Execution Framework is to replace the aging TaskMan system. However, it is not merely meant to be a simple substitute, but to expand and improve upon what TaskMan is capable of. By applying preventative measures to the developing framework, Deutsche Bank can ensure that the problems that occurred in TaskMan will not decrease JEF's lifespan. Methods that are currently utilized in the industry to avoid this type of problem have been researched and compiled into a list of best practices that Deutsche Bank should strive for.

Reactive Measures

To summarize, there are five major problems within TaskMan.

- Inability to internally utilize key, modern technologies such as secure file transfer and email.
- Restricted to a time-based scheduling system.
- Contains numerous critical points of failure.
- The jobs within TaskMan do not follow a single overall design strategy.
- Lack of personnel who understand the intricacies of TaskMan, especially those related to undocumented jobs.

It is necessary to address these concerns early, so that they do not plague the Job Execution Framework at a later date. JEF will naturally have the technologies that TaskMan lacks, but the real concern is to allow future technologies to be added to the framework as they are developed. This was solved by the departmental decision to use a service-oriented architecture design pattern, coupled with an Enterprise Service Bus for communication between services. The abstract nature of the framework's design allows for access to as-yet undeveloped technologies to be easily incorporated into the system, with minimal modification to the other services.

TaskMan's versatility is limited by its time-driven architecture. By limiting the execution of jobs to a specific set of times, many processes execute slower than the speed of business. Developing JEF as an event-driven system will alleviate this weakness, speeding up the execution of numerous processes. In a world where time is money, this is a crucial selling point. By combining this with a third-party scheduling program, such as Control-M, maximum versatility and power can be achieved, without sacrificing performance.

Internally, TaskMan is structured so that there are several points within it where the failure of a single job can cause many others to also fail. However, there is no easy cure for these single points of failure. If many jobs ultimately rely on a single, vital one, there is no simple workaround. The best defense is to identify and monitor these important processes. The TaskMan Event Calendaring System (TECS) was designed for this purpose, and will make the identification process far easier. A similar system would be useful in the upkeep and maintenance of the final JEF implementation.

While each component of TaskMan developed over the years was designed in accordance with industry standards, the resulting whole was built without an overall design schema in mind. By designing the new system as a framework, managing infrastructure is much easier, as its layered architecture isolates each service from the others, while allowing them to interact in an internally consistent manner. In turn, this will help to solve the problem of the lack of personnel who are able to sufficiently understand the job scheduling system. TaskMan's sprawling, monolithic nature effectively prevented many people from gaining an understanding of the system as a whole, limiting in-depth knowledge to a select few people.

Proactive Measures

Even if all the current problems within TaskMan are solved within the Job Execution Framework, it is not enough. A similar situation must be prevented from

necessitating a JEF replacement in the near future. In order to grant JEF the greatest possible longevity, the available technologies must be utilized in such a way that future expansion of the system can be accomplished easily. There are a number of ways to do this.

One cost-effective method is to incorporate the suggestions listed above for the Job Execution Framework's GIMS implementation. The solutions detailed are designed to eliminate the problems associated with TaskMan's GIMS implementation. New fault categories will enable support personnel to more easily identify problem areas. Coupled with a structured and informative template system, the job request system will be far more useful. By standardizing the way GIMS requests are viewed and processed, the possibilities for other future-proofing methods are expanded, as detailed below.

The TaskMan Event Calendar System will be useful in the migration of existing tasks to JEF. It has the capability to provide a highly detailed view of tasks, broken down by day and time. Its ability to export the data from a given day allows tasks to be visualized in a manner that readily displays their dependencies. This will allow for crucial processes to be easily identified, as well as for critical points of failure to be discovered.

Additionally, by having a set procedure for transferring necessary jobs from TaskMan to JEF, problems in the old application can be prevented from occurring within the new framework. Such a methodology will also ensure that the most suitable jobs are transferred first, allowing JEF a reasonable break-in period before it is running all business critical processes. In addition, having a set procedure allows for a greater degree of standardization amongst the job templates, allowing for easier maintenance and decommissioning of jobs as more and more processes are added over time.

A potentially large volume of duplicated jobs is also something that must be prevented from occurring within JEF. These jobs squander not only valuable CPU time, but the manpower required to maintain them as well. The job templates can be used to identify jobs that are based on a similar one. By including the source that a report is to be constructed from, an operator can easily determine if two reports using the same source are related. If they are, it can be flagged so that it is known which reports are based upon others. Taking such a measure will make future upkeep and maintenance of groups of reports much simpler. It will also make deletion of these reports more straightforward.

Another major problem is the large number of jobs that are no longer needed by the personnel who requested them. Under the current TaskMan GIMS, there is no way to determine this, as contact information is limited at best, and often out of date. By introducing a more robust method of maintaining usable contact data, one that includes the requestor's department so that contact can be maintained even if the requestor has since departed the company, a task recertification process can be introduced. This will involve contacting the issuing departments via e-mail at regular time intervals, such as six to twelve months. It is the goal of this process to reduce the number of unneeded jobs, reducing clutter within the JEF system, and prolonging its usable lifespan.

JEF User Stories

The following is a partial collection of user stories to be used as guidelines for what the JEF system needs to be able to do. They will be used instead of a large requirements document, in accordance with extreme programming (XP) methods. These stories have been divided into sections, based on the proposed types of classifications for JEF jobs.

File Manipulation

1. Tasks should be able to create a variety of file types (.txt, .dat, etc) and fill them with data in a specified format.

2. Files should be able to be renamed in a variety of formats (i.e. .txt to .dat).

3. Tasks should be able to check a file, change specific data within it, and save it to either its original location, or a new one.

4. Files must be able to be transferred from a machine running MS Windows, to a UNIX-based machine.

5. It should be possible for files to be encrypted, transmitted securely, and decrypted at a specified destination.

6. Tasks should be able to delete files that are outdated, or otherwise no longer needed.

SQL Procedure

1. Must be able to move external files into a database, via SQL*Loader.

2. Tasks should be able to manipulate the database through the execution of .sql scripts

3. Tasks must be able to dynamically generate SQL scripts via a variety of methods (data parsing, etc)

E-mail

1. Tasks must be able to send one or more emails at once.

2. Tasks should be able to construct email messages on the fly before sending them out to multiple addresses.

3. Email should be able to be used in a multiple part format, or display text other than standard ASCII.

Batch (or other customizable) Procedures

1. Task must be able to access and execute external programs, in order to fulfill highly customized needs.

2. Must be highly mutable to meet specific, non-standard needs.

Appendix A: Glossary of Terms

EAI	Enterprise Application Integration. An architectural principle linking together a set of enterprise computer applications. An example of an EAI is the ESB.
ESB	Enterprise Service Bus. A robust, dependable, and flexible infrastructure that can connect any IT resource, providing foundational services for more complex service oriented architectures.
FOBKIN	Futures & Options Back office Interface. Procedures consisting of code that moves data from the day-to-day RANSys database to the more persistent RANBase archive.
GES IT	Global Exchange Services Information Technology. The GES IT group provides technical development and support for the GES Front Office, Operations and BAC in Europe, the Americas, and across Asia.
GIMS	Global Incident Management System. A software program used at Deutsche Bank for logging application incidents.
JEF	Job Execution Framework. Known in its earlier stages as "TaskMan Replacement", JEF is a robust replacement for TaskMan employing SOA technologies through Sonic ESB.
RAN	Rolfe and Nolan. This company develops software specifically for the global derivatives industry. Deutsche Bank utilizes several of their software products including RANBase, RANSys, and RANTask.
SOA	Service Oriented Architecture. A software architectural concept that defines the use of intercommunicating, self contained services to work within a distributed systems architecture.
User Story	A software system requirement written on a note card (to ensure it does not grow too large), written as one or two sentences in the every day language of the user.

Table 9: Glossary of Terms

Appendix B: RANTask Task Types

The following is an alphabetical list of task types available in RANTask. This list is taken from the document “INFO – Support RANTask” by Maik Hensel.

AppendTable

Rec Builder - Appends one table in the database to another.

Archiving

Standard Functionality - Compresses files using PKzip and stores them in a specified directory.

BackupDB

Standard Functionality - Allows you to backup a database from within a job.

BusObjDocument

Business Objects - Allows you to run a Business Objects document as part of a job. This task uses Business Objects database details set up in the BusObj Domain Maintenance function .

CheckFileDate

Standard Functionality - Checks on which date a file was last modified.

CheckFileSize

Standard Functionality - Checks that the size of a file is within certain limits.

CheckLogFile

Standard Functionality - Searches a log file for a particular message.

ColManipulate

Rec Builder - Enables you to join columns, discard columns, duplicate columns or split columns in a database.

CreateColumn

Rec Builder - Adds a new column to the selected database table.

CreateTable

Rec Builder - Creates a new table in the selected database.

DropTable

Rec Builder - Deletes a table from the selected database.

Email

Emailing - Enables you to send email messages with attached files or via split report functionality.

ExecuteSQL

Rec Builder - Enables you to construct your own SQL statements which can then be executed in the selected database source.

Fax

Faxing - Enables you to send fax messages with attached files or via split report functionality.

FormatDateCol

Rec Builder - Formats a date column in the selected database table.

FormatNumCol

Rec Builder - Formats a numeric column in the selected database table. The numeric formats can be applied via a translation table. Translation tables and values are set up in Translation Maintenance.

FTP

FTP - Transfer Uploads or downloads files between your PC and a host.

Hoovering

FTP - Transfer Downloads all files for a particular date from the Rolfe and Nolan system.

ImportFile

Rec Builder - Imports a file into a new table in the selected database.

Map

Rec Builder - Applies a map to a specified table in the database.

MCMExer

MCM Functionality - Extracts each member's exercise and assignment instructions from the MCM database, creating a host file to be imported.

MCMExportTable

MCM Functionality - Exports extracted MCM data to a comma delimited file, which can then be imported by another System.

MCMGetPrices

MCM Functionality - Extracts collateral (physical) prices from the MCM database to the RANtask database.

MCMGetTheos

MCM Functionality - Extracts theoretical values and option closing prices from the MCM database to the RANtask database.

MCMRANrecFiles

MCM Functionality - Extracts open position and trade data from the MCM database to the RANtask database.

Pause

Standard Functionality - Pauses a job so that the previous task can complete, before continuing with subsequent tasks.

PCDirectoryCreate

Standard Functionality - Creates a directory on your PC or network.

PCDirectoryDelete

Standard Functionality - Deletes a specified directory from your PC or network. Only an empty directory can be deleted.

PCFileCopy

Standard Functionality - Copies one or more specified files to another location on your PC or network.

PCFileDelete

Standard Functionality - Deletes specified files from your PC or network.

PCFileExists

Standard Functionality - Searches for a specific file or a range of files on your PC or network.

PCFileMerge

Standard Functionality - Appends the contents of a specified file to the end of another. Both files must exist in the same directory but do not have to be in the same format.

PCFileRename

Standard Functionality - Changes the name(s) of one or more specified files.

PrintFile

Standard Functionality - Prints a specified file on any printer you specify.

ProgramExecution

Standard Functionality - Runs a program that is external to RANtask, using various parameters to make the task more specific.

RANwebArchive

RANweb - Transfers reports to the RANweb Archive directory so that the reports can be viewed by RANweb. The archive directory is specified in the Options window on the Locations page.

RASConnection

Dial-Up Networking - Starts a dial-up networking session with a host.

RASDisconnect

Dial-Up Networking - Closes a dial-up networking session.

Rec

Reconciliations - Applies a reconciliation to a job, which has been created via the Rec Wizard.

Reconcile

Rec Builder - Compares two tables which have a common theme and writes the results to a third table, which can then be used for reporting.

RecReport

Rec Builder - Enables you to specify the type of reports required for a reconciliation and how they are output.

ReformatMCM

MCM Functionality - Extracts each member's trades for the day from the MCM database, creating a host file to be imported by another System.

RenameColumn

Rec Builder - Renames a column in the selected table; making it numeric if required.

ReportCheck

Report Generation - Enables you to check that a specific report is available before you download it.

ReportGeneration

Report Generation - Generates a predefined report on the Rolfe and Nolan system, comprising the latest information. This task uses report batches set up in the Report Batch Maintenance function.

Reports

Standard Functionality - Enables you to upload or download reports between a host and your PC. This task uses report batches set up in the Report Batch Maintenance function.

ReportTemplate

Standard Functionality - Adds extra formatting to a report.

RescheduleJob

Standard Functionality - Reschedules a job, for example, after a convenient time delay following its failure to run successfully.

RowSelection

Reconciliations - Keeps or removes rows in a table according to the value of a particular column.

SendKeystrokes

Standard Functionality - Enables you to control programs using keystroke commands as though you had typed them on your keyboard directly.

SetColumnValue

Rec Builder - Sets a database column to a specified value. If required, this can apply only to rows where another column has a particular value.

Split

Standard Functionality - Splits certain downloaded reports using the criteria set up in Split File Maintenance and generates separate report files.

SumCols

Rec Builder - Performs mathematical operations on a database column. If required, this can apply to rows where another column has a particular value.

Supertask

Standard Functionality - Groups several tasks within a job.

TextTranslator

Text Translator - Searches and replaces text within a specified file or group of files.

TransferData

Data Access - Transfers selected data from one type of data source to another (including to a file).

TranslateCols

Rec Builder - Translates the values in a database table according to a specified value in a pre-defined translation table. Translation tables and values are set up in the Translation Maintenance function.

Translation

Standard Functionality - Replaces coded fields in a report with addresses stored locally in the Address Maintenance function. Some European countries have a regulatory requirement whereby the address of a bank or broker must not be stored on a computer of another company or in another country.

TrimCols

Rec Builder - Discards (trims) a number of characters from a database column. If required, leading or trailing spaces can also be removed.

UnixDosConverter

Standard Functionality - Converts an imported UNIX file to DOS format, ready for processing for a reconciliation.

WRQTransfer

WRQ Transfer - Uploads or downloads reports between a host and your PC via Reflection. This task uses report batches set up in the Report Batch Maintenance function.

Appendix C: TaskMan Job Types

BSR: Bank Statistical Reviews
Generates statistical reviews for the Bank of England.

Bulk File Import
Used to copy multiple files within one task.

Capital Adjustments
Populates a table with capital adjustment information.

Cash Settlements
Populates a table with cash settlements information.

Commission
Populates a table with commission information.

Convert File Format
Reads in a file, performs a custom transformation, and rewrites the file to a new location.

Database Table Archive
Archives a table to file.

Event Monitor
A task to monitor for sql ".bad" files.

Exercise Premium
Populates a table with premium information.

Export Mappings
Export mappings from a database.

FFTFAX: Frankfurt Fax
Parses a file for sending a facsimile, usually via ftp.

File Import
Copies a file from a specified location.

File Import 2
Copies a file from a specified location, also allows for the filename of the imported file to be dynamically determined based off of a sequence number.

FOGL: Futures & Options General Ledger

Creates the DMG individual posts general ledger.

GENLED: General Ledger

Creates a report displaying the general ledger.

Initial Margin

Populates a table with initial margin information.

Intermeadiate Margin

Generates a temporary file used in calculating an initial margin.

P&L Premium

Populates a table with premium information for P&L.

PalRep: Profits and Losses Report

Generates a report for profits and losses.

Premium

Populates a table with premium information.

Profit & Loss

Populates a table with profits and losses.

Replix FAX

Parses a file for sending a facsimile using Replix, a third party fax server solution.

Run SQL File

Execute a SQL file against a database server.

Squash / Difference GENLED Files

This procedure will first "squash" (i.e. merge) all of the similar account transactions in a GENLED file that can be represented individually. It then performs a difference check against a previous GENLED file.

Stored Procedure

The Stored Procedure type provides a simple way to perform a call on an external batch file. Any operation needing to be implemented that is not currently supported by TaskMan is represented as a stored procedure. In doing this Deutsche Bank was able to effectively freeze development of TaskMan while still being able to meet the requirements of newer complex scheduling jobs.

Variation Margin

Populates a table with variation margins.

Volume Reports (daily / monthly)

Creates weekly and monthly volume reports.

Appendix D: Task Discrepancies

The following is a list of TaskMan jobs that appear to wait on tasks that do not exist, or are no longer running (as of 2005-11-17).

TaskAIB_w2k.export	'HOSTTRNAY Load' waits on non-existent task 'HOSTTRNAY Import'
TaskGenUS_w2k.export	'UploadProblemMail' waits on non-existent task 'GENLEDC2 UPLOAD (US)'
TaskGenUS_w2k.export	'UploadProblemMail' waits on non-existent task 'GENLEDC3 UPLOAD (US)'
TaskGenUS_w2k.export	'UploadProblemMail' waits on non-existent task 'GENLEDC4 UPLOAD (US)'
TaskGenUS_w2k.export	'UploadProblemMail' waits on non-existent task 'GENLEDC5 UPLOAD (US)'
TaskGenUS_w2k.export	'UploadProblemMail' waits on non-existent task 'GENLEDC7 UPLOAD (US)'
TaskGenUS_w2k.export	'UploadProblemMail' waits on non-existent task 'GENLEDCL UPLOAD (US)'
TaskMisc2_w2k.export	'HOSTBALS5 US Upload' waits on non-existent task 'HOSTBALS5 Delete Scratchpad'
TaskSAMBA_w2k.export	'COMMODITIES CRAC trades feeder' waits on non-existent task 'COMMODITIES CRAC'
TaskUAT_w2k.export	'HOSTINF Upload ACS' waits on non-existent task 'HOSTINF Import US'
TaskUAT_w2k.export	'UAT COMMODITIES CRAC trades feeder' waits on non-existent task 'UAT COMMODITIES CRAC'
TaskUAT_w2k.export	'UAT DREX COMMODITIES trades feeder' waits on non-existent task 'UAT DREX COMMODITIES'
TaskUAT_w2k.export	'UAT DREX IMAGINE trades feeder' waits on non-existent task 'UAT DREX IMAGINE'
TaskUAT_w2k.export	'UAT DREX STARS trades feeder' waits on non-existent task 'UAT DREX STARS'
TaskUAT_w2k.export	'UAT LDN HostINF upload ACS' waits on non-existent task 'UAT LDN HOSTINF Import'
TaskUAT_w2k.export	'UAT LDN HostINF upload ACS' waits on non-existent task 'UAT LDN PRICES Import'
TaskUAT_w2k.export	'UAT LDN HostINF upload ACS' waits on non-existent task 'UAT LDN RECAP Import'
TaskUSAClient_w2k.export	'GlobeOp Intraday Reset' waits on non-existent task 'GlobeOp Intraday FTP'

Appendix E: Included Media

Included with this report is a CD or other form of readable digital media containing supplementary content. The data provided was either not feasible to include in the body of this report, or is included but may be more useful in digital form.

Included directory structure

/

DIR.txt – This directory structure.

/doc_templates

gims.doc – Sample Microsoft Word document including forms for making GIMS requests.

/jobs

all_jobs_24.10.2005.xls – Excel workbook listing all jobs in the system as of 24.10.2005.

complete_job_distributions.xls – Distribution of all (enabled and disabled) TaskMan job types.

discrepancies.txt – Generalization of job discrepancies.

discrepancies_detail.xls – Excel workbook detailing job discrepancies.

gims_q3_2005.xls – Excel workbook listing all GIMS requests for the third quarter of 2005 and referencing them to an established "Basic Function".

job_types.xls – Excel workbook listing all TaskMan and RANTask job types, as well as their respective functionality.

stored_procedrue_analysis.xls – Analysis of all stored procedures in the TaskMan system.

task_errors.txt – Text documenting listing specific task discrepancies in the TaskMan system.

taskman_requests.txt – Text document breaking down the findings of 3rd Quarter 2005.xls.

/misc

jef_diagram.vsd – Visio diagram displaying the layout of the JEF system.

/scripts

tecs-mysql.sql – MySQL script for creating an appropriate schema and table structure for TECS.

tm2csv.pl – Perl script for converting TaskMan export files into a CSV representation.

tm2db.pl – Perl script for populating a MySQL database with data from TaskMan export files.

tm2xls.pl – Perl script for converting TECS xml files into an XLS representation.

/scripts/include

DBI-1.49.tar.gz – Database independent interface used for all database interaction in included pl scripts. This is version 1.49 obtained from CPAN on 13/12/2005.

Spreadsheet-WriteExcel-2.15.tar.gz – WriteExcel Module used in tm2xls.pl. This module is version 2.15 obtained from CPAN on 13/12/2005.

XML-Parser-2.34.tar.gz – XML::Parser Module used in tm2xls.pl. This module is version 2.34 obtained from CPAN on 13/12/2005.

/tecs

README – Installation information and release notes for the TaskMan Event Calendaring System.

tecs-1.0.tar.gz – The TaskMan Event Calendaring System.

tecs-1.0.vsd – Class diagram for the TaskMan Event Calendaring System.

Appendix F: Works Cited

BMC Software. (2001). *Business Integrated Scheduling*. BMC Software Onsite.

Craggs, Steve. (2003). *Best-of-Breed EABs: Identifying best-of-breed characteristics in Enterprise Services Buses (ESBs).* EAI Industry Consortium.

Deutsche Bank. (2005). Deutsche Bank at a Glance. Retrieved 20 September, 2005 from http://www.db.com/en/content/company/our_company.htm

Dumas, Jonathan et al. (2004). *Business Rule Creation for Deutsche Bank's Dashboard program.* Worcester Polytechnic Institute. 04B028M.

GES-IT. (2003). *Current System Description.* Deutsche Bank.

OBI Support Group. (2005). *TaskMan Application Support*. Deutsche Bank.

Operations, BAC, & Infrastructure IT. (2005). *Project Initiation Document – TaskMan Replacement*. Deutsche Bank.

Rolfe and Nolan. (2005). *RANsys*. Rolfe and Nolan.

Sonic Software. (2005). *Datasheet: Sonic ESB.* Sonic Software Corporation.

Sonic Software. (2003). *Whitepaper: Distributed Service-Oriented Architectures.* Sonic Software Corporation.

Veale, Stuart. (2001). *Stocks Bonds Options Futures*. Prentice Hall Press.